Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина: Программирование (КПиЯП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

Реализация тотализатора результатов матчей

БГУИР КП 1-53 01 07 10724 ПЗ

Студент: гр. 253505 Волчецкий А. М.

Руководитель: ассистент кафедры информатики Шиманский Валерий Владимирович

Минск 2013

Содержание

Введение…………………………………………………………………………3

1. Анализ предметной области…………………………………………………4

1.1 Теория…………………………………………………………………4

1.2. Анализ существующих аналогов…………………………………...6

1.3 Постановка задачи……………………………………………………8

2. Разработка программы для осуществления ставок на спорт..…………….9

2.1 Виды реализуемых ставок…...………………………………………9

2.2. Требования к программе……………………………………………9

3.Методика работы с полученной программой………………………………10

Заключение……………………………………………………………………...15

Литература………………………………………………………………………17

Приложение 1. Исходный код………………………………………………….19

Введение

Букмекерские ставки являются любимым видом развлечений и азартных игр, корни которых достигают начала истории человечества. Первые упоминания о заключении пари на деньги относится к Лондону XVIII-го века, где в аукционном доме Ричарда Таттерсалла начали принимать ставки на лошадиные бега. Термин букмекерство в этом контексте означало ведение книги записей всех принятых ставок.

Именно в Англии еще в начале ХХ века появилось понятие „букмекер” и были созданы первые легальные официальные пункты приема ставок на спорт. Человеком, который ввел моду на букмекерские конторы, был Джон Мур, который открыл первую контору в 1922 году в Ливерпуле „Littlewood Pools of Liverpool”. Само понятие состоит из двух слов - „book” (ставка) и „maker”(делать).

В связи с этим на ум приходит вопрос: откуда пришла идея принимать ставки? Ответ на этот вопрос состоит из двух частей. Конец ХIХ и начало ХХ веков считается периодом развития наиболее популярной спортивной дисциплины — футбола. В Англии в 1888 году начались первые в мире профессиональные матчи. Заинтересованность этим видом спорта очень быстро росла, особенно благодаря региональным журналистским изданиям, которые представляя обзоры матчей английской лиги начинали делать свои прогнозы на следующие матчи и даже возможные коэффициенты на игру. Другим поводом конечно были деньги. Одна сторона заключающая пари всегда несет риск поражения, независимо от исхода игры.

Растущая популярность контор не могла пройти без эха. К сожалению развитие пунктов приема ставок привело к тому, что доходы букмекерских контор сильно упали. Рынок был переполнен конторами, новых клиентов не прибывало, а букмекеров наоборот становилось все больше и больше. Спортивные ставки и целый рынок связанный с ними, перестал быть приемлемым и симпатичным способом заработка легких денег.

Важным шагом для выхода из кризиса стало распространение деятельности букмекеров в Интернете. Эта идея появилась у американцев, которые желали заработать на букмекерских ставках. Развитие телекоммуникации и информатики стало поводом того, что игроки имели неограниченные возможности размещения ставок. Главной проблемой было малое количество игроков, которые имели компьютеры. Первая официальная ставка сделанная в сети датируется на 1972 год в Канаде. Ставки в интернете давали игрокам возможность создания букмекерских купонов в любое время суток, в любой точке мира. Единственным условием было владение компьютером или мобильным телефоном и доступ к Интернету. Эти условия были выполнены в глобальном масштабе только в середине 90-х годов ХХ века. В те времена проходил настоящий бум букмекерских контор. На сегодняшний день на рынке действует несколько десятков букмекеров онлайн.

1. Анализ предметной области

1.1 Теория

**Термины**

**Букмекер** (англ. *bookmaker*) — профессия профессионального спорщика, занятого приёмом денежных ставок на различные предстоящие события (чаще всего спортивные) с заранее оговоренными выигрышами, а также выплатой выигрышей. Букмекеры принимают ставки при игре на скачках и бегах, на результаты различных спортивных, политических, культурных, финансовых, погодных и прочих событий.

**Спортивный беттинг** — ставки в букмекерских конторах на результаты спортивных состязаний. В спортивном беттинге распространены ставки не на вероятность конечного результата, а на ожидаемую разницу в счете. Изобретателем этой системы был **Чарльз К. Макнейл** (1903–1981), преподаватель математики из Коннектикута.

**Беттор** (англ. *bettor*) — человек, делающий ставки.

**Классификация ставок**

#### ****По важности****

* Ставки на основные исходы (match bets):  
  победа участника 1, победа участника 2, ничья, их производные фора и тотал (общее количество очков) в матче
* Ставки на второстепенные исходы:

промежуточные итоги события, кто забьет, количество удалений, количество угловых и любые другие характеристики события, вероятность которых можно оценить

#### ****По количеству исходов****

* Двух-исходные: победа участника 1 или победа участника 2
* Трех-исходные: победа участника 1, победа участника 2, или ничья
* Многоисходные: ставки на события с количеством участников более трех или на второстепенные исходы

#### ****По сложности****

* Простые ставки: победа участника 1, победа участника 2, или ничья
* Ставки с форами (spreads), или гандикапы: победа команды 1 с разницей в определенное количество очков, поражение команды 2 с разницей в определенное количество очков

#### ****По составу****

* Одиночные (ординары): на один исход одного события
* Комбинации (экспрессы и системы): одновременно на несколько исходов нескольких спортивных событий

#### ****По отношению к времени проведения события****

* Долгосрочные (futures): на победу в чемпионате при его открытии
* Краткосрочные: на текущие события, принимаются до начала события
* В процессе игры (live)

**Подробнее об основных видах ставок**

**Ординар:** ставка делается на исход одного события. Можно ставить на любой исход: победа команды, ничья, точный результат матча, общее количество голов и т.д. Возможный выигрыш равен произведению суммы денег, поставленных на матч, и итогового коэффициента ставки. Это правило применимо ко всем видам ставок.

**Экспресс:** ставка похожа на ординар. Одно лишь отличие в том, что здесь выбираются 2 и более событий. При этом итоговый коэффициент будет равен произведению коэффициентов всех исходов ставки. Ограничение на максимальное количество событий ставит букмекерская контора.

**Система**: самый сложный вид ставок, но в то же время самый безопасный для беттора. Также, как и в экспресс-ставках, выбирается 2 и более события. Дополнительно указывается минимум результатов, которые беттор должен угадать для получения выигрыша. Например, если выбрана ставка размерностью 2/3 (2 события из 3-ех), то даже в случае неправильно угаданного результата одного события, беттор все равно сможет получить выигрыш (конечно, при условии что 2 события он все-таки угадает). Правда в таком случае выигрыш обычно получается меньше, чем сумма ставки. Формула расчёта возможного выигрыша здесь сложнее. Например, беттор поставил на всю туже систему размерностью 2/3 3$. А коэффициенты на события будут такими: на событие x коэффициент 2, на событие y—1,5, на событие z—4. Далее сумма денег делится на количество выбранных событий, а полученное число считается ставкой на отдельное событие. В нашем случае сумма m=1$ (3$/3=1). Затем выбираются все возможные комбинации, состоящие из 2-ух матчей (т.к. у нас система 2/3). Это будут комбинации: xy, xz, yz. В итоге получаем формулу для расчета возможного выигрыша r = m\*x\*y + m\*x\*z + m\*y\*z. Далее подставляем вместо переменных нужные коэффициенты и находим выигрыш. Если ставка на одно событие не сыграла, то коэффициент на него становится равным 0. В примере выигрыш при совпадении всех 3-ех вариантов будет равен 17$, при совпадении 2-ух(ставка на событие z не сыграла)—3$, при совпадении 1-го(ставки y и z не сыграли)—0$.

**Фора:**  преимущество или отставание команды (игрока, гонщика и т.п.), выраженное в голах, очках, сетах, секундах и т.д., которое определяется букмекерской конторой по конкретной ставке.  
Исход события с учётом форы определяется прибавлением форы к фактическому результату. Если полученный результат в пользу выбранной команды (игрока, гонщика и т.п.), ставка считается выигранной. Если наоборот, то ставка считается проигранной. Если полученный с учётом форы результат – ничья, то коэффициент выигрыша по такому исходу будет равен «1».

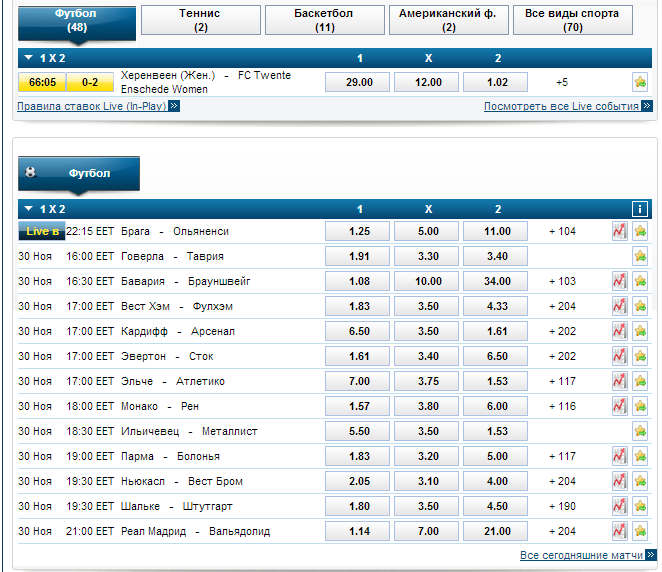
**Тотал**: Ставка на общее количество голов, очков, геймов и т.д., забитых, набранных, сыгранных и т.д. командами (игроками и т.д.). Для выигрыша необходимо определить, сколько будет забито, набрано, проведено больше, меньше данного в линии тотала. Букмекерская контора предлагает два варианта пари на тотал – по двум («меньше» или «больше»). Если результат совпадёт с предложенным букмекерской конторой значением тотала, коэффициенты выигрышей по ставкам на «меньше» или «больше» считаются равными «1».

1.2. Анализ существующих аналогов

Т.к. сегодня многие букмекерские конторы имеют свои сайты в интернете, то и соответственно ставки делают у них на сайте. Надобности в написании отдельных программ нет. Рассмотрим сайты наиболее популярных букмекерских контор.

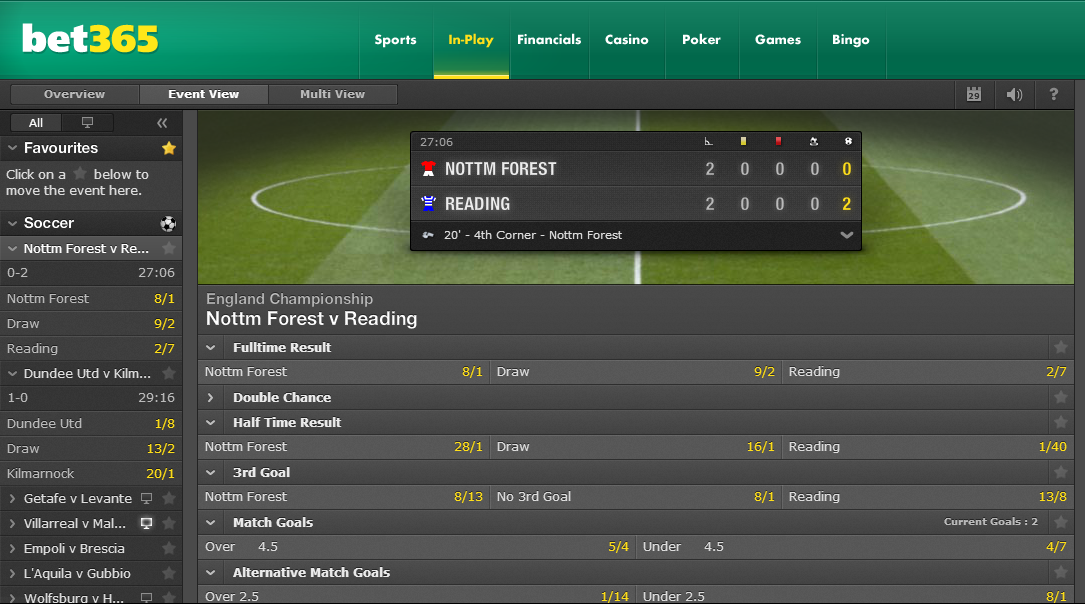
***Первый сайт*** [http://sports.williamhill.com](http://sports.williamhill.com/)***.***

Интерфейс выглядит так:

**

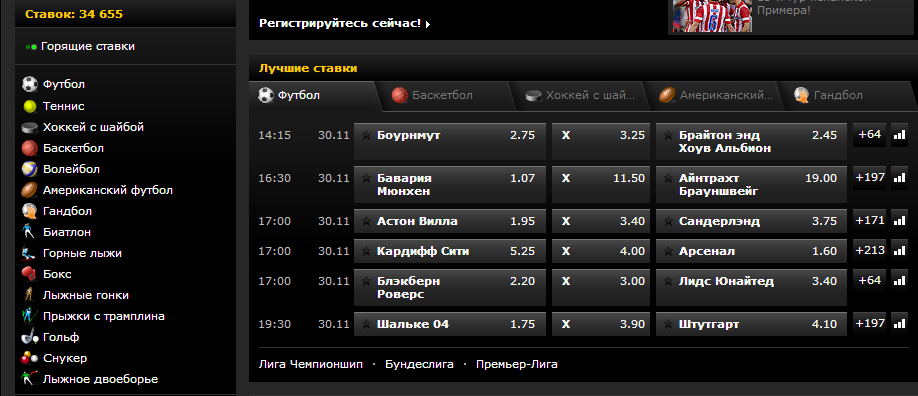
***Второй сайт*** [http://www.bet365.com](http://www.bet365.com/)

Интерфейс выглядит так:

******

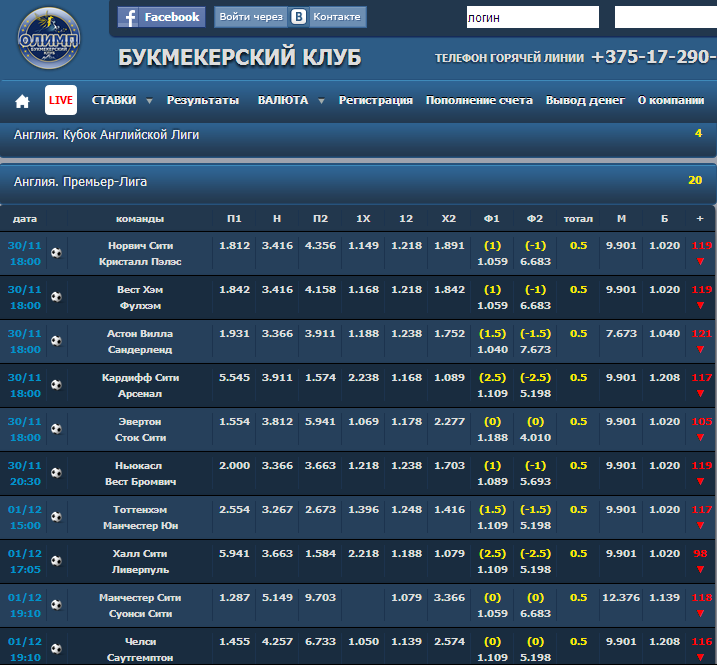
***Третий сайт*** <https://sports.bwin.com/ru/sports>

Интерфейс выглядит так:



***Четвертый сайт (белорусский аналог)*** [http://olympbet.com](http://olympbet.com/)

Интерфейс выглядит так:



1.3 Постановка задачи

В задачу курсового проекта входит:

* 1. Разработка программного средства на языке C++, дающего возможность пользователю выполнять ставки на результаты спортивных матчей.
  2. Программное средство должно выполнять следующие функции:
     + Осуществление ставок на результаты спортивных матчей пользователем
     + Управление пользователем своим денежным балансом
     + Представление статистики по всей букмекерской конторе
  3. Реализация понятного и удобного для пользователя интерфейса программного средства.

Таким образом, задача данного курсового проекта сводится к разработке программного средства, предоставляющего пользователю возможность делать ставки на разные виды спорта, отображая при этом возможный выигрыш, а также обеспечивает работу с денежными средствами, а именно: вывод и ввод денег в любой валюте.

2. Разработка программы для осуществления ставок

2.1 Виды реализуемых ставок

В программе планируется реализовать следующие виды ставок:

* Ординар
* Экспресс
* Система

2.2 Требования к программе для осуществления ставок

Необходимо написать программу на С++, которая смогла бы осуществлять ставки и отображать ставки на экране. В связи с этим к программе предъявляются следующие требования:

1. Интерфейс.

Интерфейс программного средства будет содержать рабочую область, в которой и будут отображаться матчи проходящие в выбранный день. Так же должно быть меню, в котором будут находиться пункты, позволяющие продемонстрировать осуществленные ставки пользователя, а также его деньги в букмекерской конторе. Так же при добавлении/удалении ставок пользователь сможет видеть текущий общий коэффициент и текущий возможный выигрыш.

1. Функциональность.

Данная программа должна выполнять следующие требования:

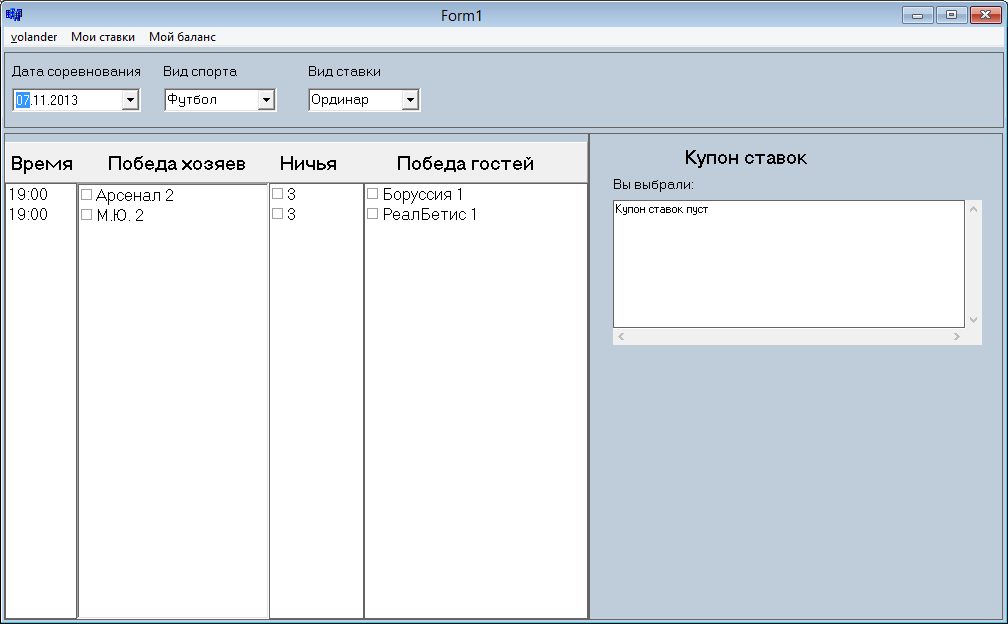
* + Предоставлять пользователю выбор вида ставок
  + Во время выбора событий показывать промежуточные результаты коэффициентов и выигрыша
  + Иметь возможность очистить рабочую область
  + Возможность регистрации нового пользователя

1. Оптимальность используемых алгоритмов. Все используемые алгоритмы протестированы на оптимальность, дальнейшее их улучшение не имеет смысла.
2. Совместимость. Данное программное средство совместимо с операционными системами семейства Windows, такими как XP и Windows7.
3. Среда разработки. Средой разработки данного программного средства выбран C++ Buider 6 от Borland. Выбор остановился на этой среде разработки потому, как в ней реализовано множество полезных классов, как для работы с графикой, так и для работы с формой и меню. В данном проекте нам пригодятся классы TMainMenu – отвечает за отображение меню и обработку нажатий клавиш меню, класс DateTimePicker—предоставляет возможность выбора даты соревнования, класс TCheckListBox—выбор результатов событий для ставки, выбор валюты для ввода/вывода денег, класс TComboBox для выбора валюты для суммы денег, которая будет поставлена на результат события, для выбора вида ставок, выбора вида спорта.

3. Методика работы с полученной программой

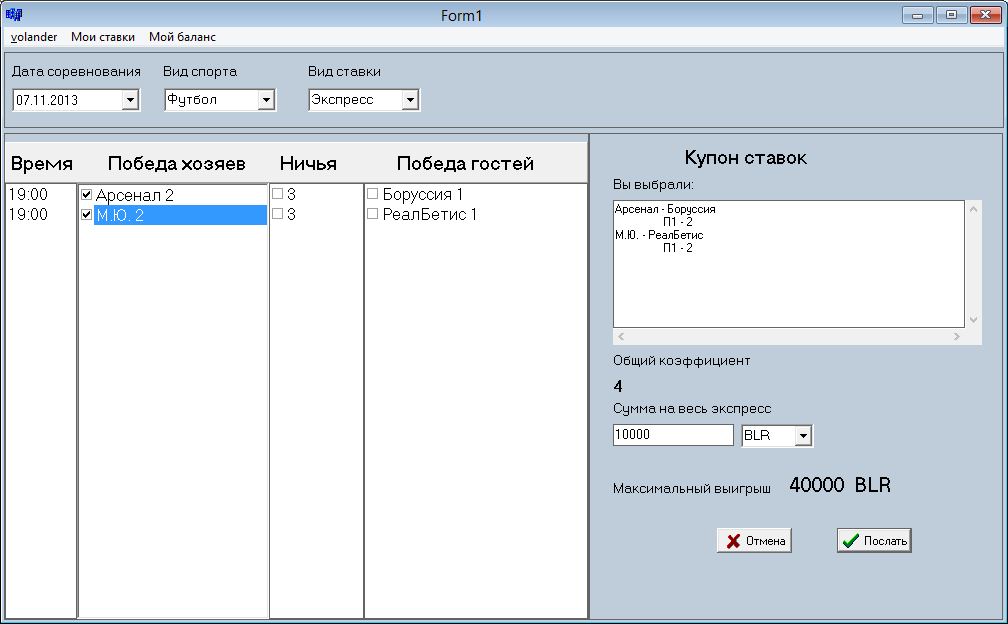
Интерфейс полученной программы состоит из Меню, и рабочей области, в которой выводятся матчи, которые будут происходить в выбранный день.

Основное меню выбора матчей:



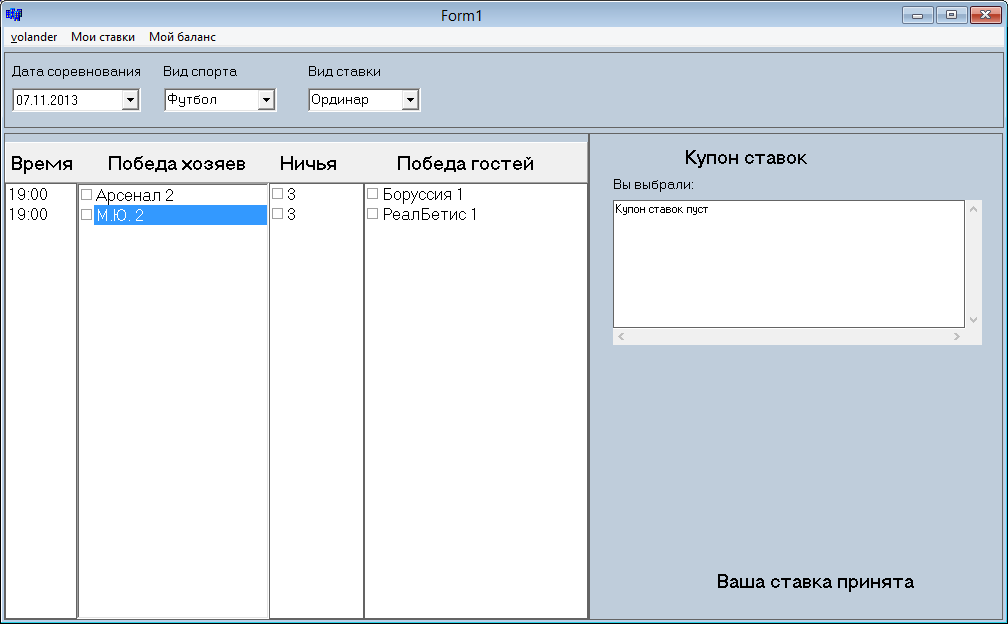
Продемонстрируем работу с программой:

Выберем ставку экспресс. Поставим на победу команды Арсенал и М.Ю 10000 бел. рублей. Мы увидим на экране общий коэффициент ставки и максимальную сумму выигрыша

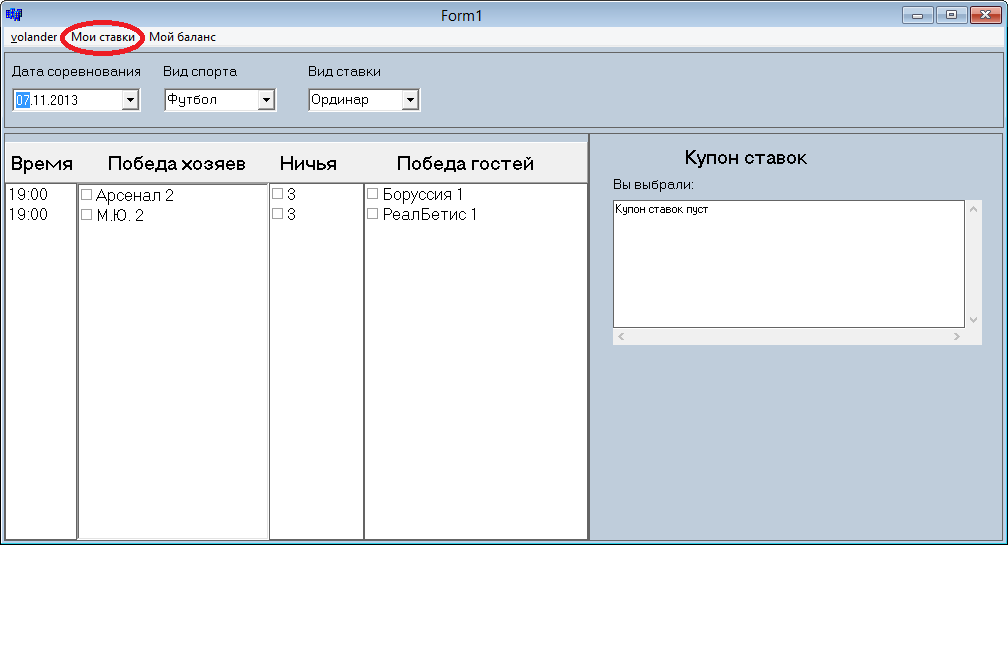


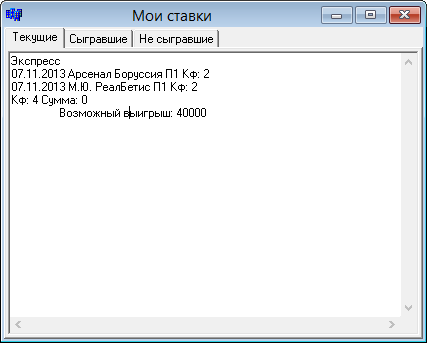
Нажмем кнопку «Послать», для того чтобы наша ставка была принята.

Мы увидим окно:



Это значит, что наша ставка успешно осуществлена.

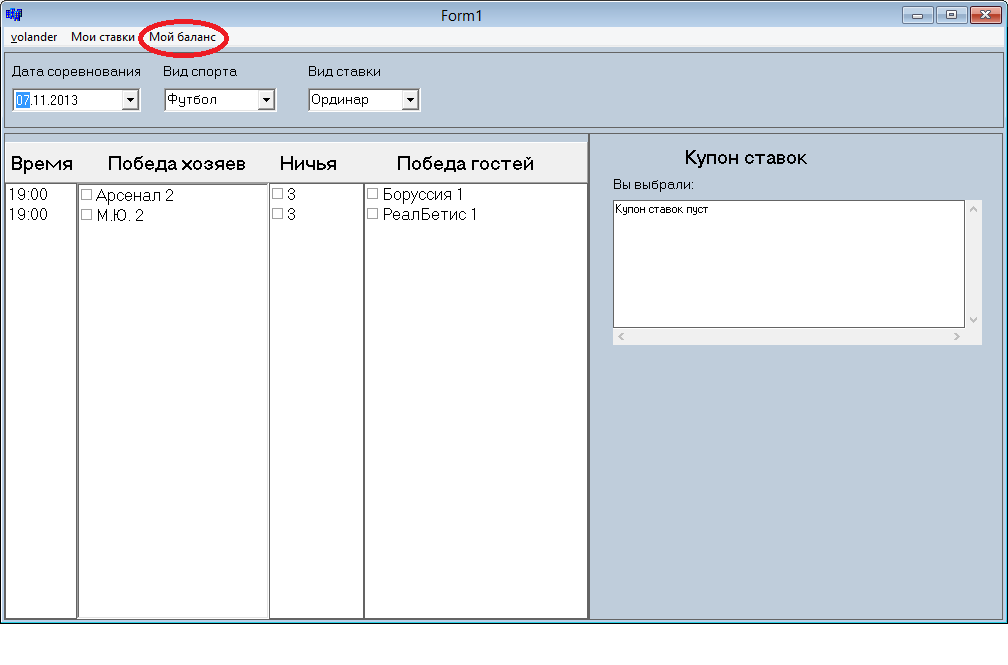
Для того, чтобы просмотреть текущий статус ставки , нажмем кнопку «Мои ставки» класса TMainMenu Появится окно, в котором можно увидеть текущие, сыгравшие и не сыгравшие ставки



Аналогичным образом работает и осуществление других ставок.

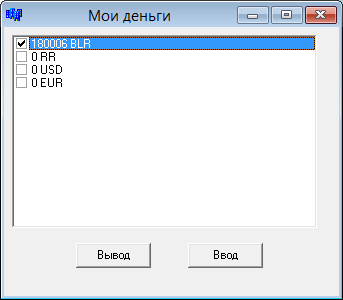
Продемонстрируем работу программы с деньгами.

При нажатии кнопки класса TMainMenu «Мой баланс»

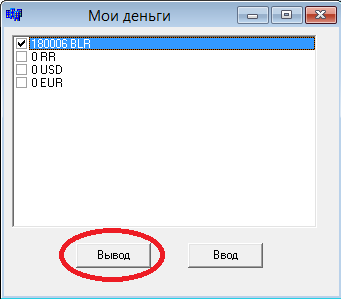


Появится окно «Мои деньги», где можно пополнить счет, либо вывести деньги со счета. Продемонстрируем работу вывода денег с лицевого счета. Для этого нужно:

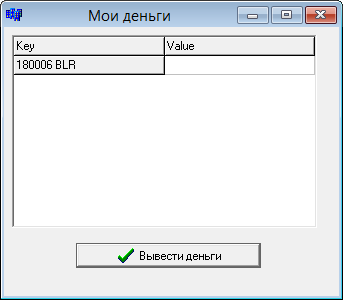
Выделить нужную(ые) валюту(ы):



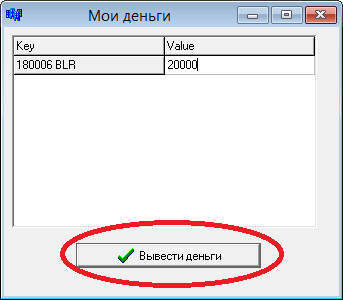
Нажать кнопку «Вывод»



Далее появится окно:

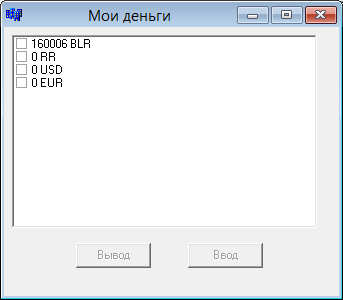


Здесь указывается размер выводимой суммы денег. Укажем для примера 20000 BLR и нажмем кнопку вывести деньги:



Деньги перевелись кошелек WebMoney, который был указан при регистрации.

Снова нажмем кнопку «Мой баланс». Как видно, баланс в бел. рублях уменьшился на 20000. Что и следовало ожидать.



Заключение

В результате работы над курсовым проектом было разработано программное средство для осуществления ставок на результаты спортивных матчей.

В ходе работы над проектом было изучено много информации по теме ставок и букмекерских контор. Были рассмотрены различные виды ставок, и выбраны наиболее популярные и известные из них.

Для каждого вида ставок был создан отдельный список, который содержал всю нужную информацию для о ставке и событии.

Так же были рассмотрены аналоги данной программы, а именно, сайты:[http://sports.williamhill.com](http://sports.williamhill.com/)***,*** [http://www.bet365.com](http://www.bet365.com/), <https://sports.bwin.com/ru/sports>***,*** [http://olympbet.com](http://olympbet.com/). В программе реализован понятный пользователю интерфейс, возможность выбора вида ставок из списка.

Были изучены возможности работы с различными компонентами в среде С++, а именно работа с компонентами среды разработки C++ Builder от Borland.

Разработанное программное средство представляет собой законченный продукт, готовый к использованию. Однако при желании функционал программы можно расширить, добавив новые функции.

Список использованных источников

1. Сайт <http://allsportlinks.com/> - теория
2. Сайт <http://www.stavki.info/> - теория
3. Сайты [http://sports.williamhill.com](http://sports.williamhill.com/)***,*** [http://www.bet365.com](http://www.bet365.com/), <https://sports.bwin.com/ru/sports>***,*** [http://olympbet.com](http://olympbet.com/) - букмекерские конторы

Приложение 1. Исходный код

//---------------------------------------------------------------------------

#ifndef Unit1H

#define Unit1H

//---------------------------------------------------------------------------

class TForm1 : public TForm

{

\_\_published: // IDE-managed Components

TGroupBox \*GroupBox1;

TDateTimePicker \*DateTimePicker1;

TLabel \*Label1;

TComboBox \*choiceSport;

TLabel \*Label2;

TComboBox \*choiceBet;

TLabel \*Label3;

TGroupBox \*GroupBox2;

TCheckListBox \*homeCB;

TCheckListBox \*drawCB;

TListBox \*timeLB;

TCheckListBox \*guestCB;

TGroupBox \*GroupBox3;

TLabel \*Label4;

TLabel \*Label5;

TMemo \*Memo1;

TLabel \*Label6;

TEdit \*Edit1;

TComboBox \*choiceCurrencyCB;

TLabel \*Label7;

TLabel \*Label8;

TLabel \*Label9;

TLabel \*Label10;

TEdit \*Edit2;

TLabel \*Label11;

TLabel \*Label12;

TLabel \*Label13;

TLabel \*Label14;

TBitBtn \*BitBtn1;

TBitBtn \*BitBtn2;

TLabel \*Label15;

TMainMenu \*MainMenu1;

TMenuItem \*N1;

TMenuItem \*N2;

TMenuItem \*N4;

TLabel \*Label16;

TLabel \*Label17;

TMenuItem \*userName1;

TMenuItem \*N3;

TMenuItem \*N5;

TPanel \*Panel1;

TLabel \*Label18;

void \_\_fastcall FormCreate(TObject \*Sender);

void \_\_fastcall DateTimePicker1Change(TObject \*Sender);

void \_\_fastcall choiceSportChange(TObject \*Sender);

void \_\_fastcall drawCBClick(TObject \*Sender);

void \_\_fastcall guestCBClick(TObject \*Sender);

void \_\_fastcall Edit1Change(TObject \*Sender);

void \_\_fastcall choiceBetChange(TObject \*Sender);

void \_\_fastcall Edit1Click(TObject \*Sender);

void \_\_fastcall choiceCurrencyCBChange(TObject \*Sender);

void \_\_fastcall homeCBClickCheck(TObject \*Sender);

void \_\_fastcall BitBtn1Click(TObject \*Sender);

void \_\_fastcall BitBtn2Click(TObject \*Sender);

void \_\_fastcall N2Click(TObject \*Sender);

void \_\_fastcall N1Click(TObject \*Sender);

void \_\_fastcall Edit2Change(TObject \*Sender);

void \_\_fastcall Edit2Click(TObject \*Sender);

void \_\_fastcall N4Click(TObject \*Sender);

void \_\_fastcall N3Click(TObject \*Sender);

void \_\_fastcall N5Click(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TForm1(TComponent\* Owner);

void MathesToForm(list<Match> todayMathes);

void ChoiceMatch(TCheckListBox \*checkBox, list<Bet>& betList);

void MakeBet(TCheckListBox \*checkBox);

void AllUnChecked(TCheckListBox \*checkBox);

void AddToMemo(string teamAndKoef);

void DeleteFromMemo(string line);

void ClearForm();

void BetToForm();

bool isInitialize;

void SaveBetsToFile(string path);

void MyBetToForm(ifstream& file, ofstream& fileCopy);

void MyMoneyToForm();

void NotEnter();

void Enter();

double minBet();

bool isEnoughMoney(double moneyBet);

};

//---------------------------------------------------------------------------

extern PACKAGE TForm1 \*Form1;

//---------------------------------------------------------------------------

#endif

void \_\_fastcall TForm1::FormCreate(TObject \*Sender)

{

MainMenu1->Items->Count;

isInitialize = false;

choiceSport->ItemIndex = 0;

choiceBet->ItemIndex = 0;

choiceCurrencyCB->ItemIndex = 0;

Edit2->Text = "0";

Edit1->Text = "1";

Edit2->Hide();

Label7->Hide();

Label11->Caption = "/0";

isInitialize = true;

Edit1->Text = "0";

Label11->Hide();

Label12->Hide();

Label13->Hide();

Label15->Hide();

Label16->Hide();

Label17->Hide();

BitBtn1->Hide();

BitBtn2->Hide();

Label18->Hide();

Label6->Hide();

Edit1->Hide();

choiceCurrencyCB->Hide();

Label7->Hide();

Label14->Hide();

Label10->Hide();

Label8->Hide();

Label9->Hide();

LoadSports();

//DateTimePicker1->Date = Date(); óñòàíàâëèâàåò òåêóùóþ äàòó

string currentDate = DateToStr(DateTimePicker1->Date).c\_str();

list<Match> todayMathes = TodayMathes(football, currentDate);

MathesToForm(todayMathes);

//LoadMoneyFromFile();

}

//---------------------------------------------------------------------------

void TForm1::MathesToForm(list<Match> todayMatches)

{

int size = todayMatches.size();

for(int i = 0; i < size; i++)

{

Match temp = todayMatches.front();

todayMatches.pop\_front();

timeLB->Items->Add(temp.time.c\_str());

string teamAndKoef = temp.home + " " + (IntToStr(temp.winnerHome)).c\_str();

homeCB->Items->Add(teamAndKoef.c\_str());

drawCB->Items->Add(IntToStr(temp.draw));

teamAndKoef = temp.guest + " " + (IntToStr(temp.winnerGuest)).c\_str();

guestCB->Items->Add(teamAndKoef.c\_str());

}

}

void \_\_fastcall TForm1::DateTimePicker1Change(TObject \*Sender)

{

Form3->Memo1->Clear();

Form3->Memo2->Clear();

Form3->Memo3->Clear();

timeLB->Items->Clear();

homeCB->Items->Clear();

drawCB->Items->Clear();

guestCB->Items->Clear();

list<Match> sport = ChoiceSport(choiceSport);

string currencyData = DateToStr(DateTimePicker1->Date).c\_str();

list<Match> todayMathes = TodayMathes(sport, currencyData);

MathesToForm(todayMathes);

Label15->Hide();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::choiceSportChange(TObject \*Sender)

{

timeLB->Items->Clear();

homeCB->Items->Clear();

drawCB->Items->Clear();

guestCB->Items->Clear();

list<Match> sport = ChoiceSport(choiceSport);

string currencyData = DateToStr(DateTimePicker1->Date).c\_str();

list<Match> todayMathes = TodayMathes(sport, currencyData);

MathesToForm(todayMathes);

Label15->Hide();

}

//---------------------------------------------------------------------------

void TForm1::ChoiceMatch(TCheckListBox \*checkBox, List& betList)

{

int number = checkBox->ItemIndex;

string teamAndKoef = (checkBox->Items->operator [](number)).c\_str();

list<string> temp = Parse(teamAndKoef);

string selectedTeam = temp.front();

string resultMatch = "";

float koef = StrToFloat(temp.back().c\_str());

string drawResult = temp.back(); //åñëè âûáðàíà ñòàâêà íà íè÷üþ, òî

//ôóíêöèÿ Parse() âîçâðàòèò ñïèñîê èç îäíîãî ýëåìåíòà

//òàì áóäåò òîëüêî çíà÷åíèå êîýôôèöèåíòà

if(drawResult == selectedTeam)

{

resultMatch = "Í";

}

teamAndKoef = (homeCB->Items->operator [](number)).c\_str();

temp = Parse(teamAndKoef);

string teamHome = temp.front();

if (teamHome == selectedTeam)

{

resultMatch = "Ï1";

}

teamAndKoef = (guestCB->Items->operator [](number)).c\_str();

temp = Parse(teamAndKoef);

string teamGuest = temp.front();

if (teamGuest == selectedTeam)

{

resultMatch = "Ï2";

}

string toMemo = teamHome + " - " + teamGuest + "\n\t" + resultMatch + " - " + (FloatToStr(koef)).c\_str();

string date = DateToStr(DateTimePicker1->Date).c\_str();

string time = (timeLB->Items->operator [](number)).c\_str();

if(checkBox->Checked[number])

{

AddBetList(betList, resultMatch, koef, date, time, teamHome, teamGuest);

AddToMemo(toMemo);

BitBtn1->Show();

BitBtn2->Show();

Label6->Show();

Edit1->Show();

choiceCurrencyCB->Show();

Label8->Show();

Label9->Show();

}

else

{

DeleteBetList(betList, date, time, teamHome, teamGuest, resultMatch);

DeleteFromMemo(toMemo);

if (Memo1->Lines->operator [](0) == emptyCoupon.c\_str())

{

BitBtn1->Hide();

BitBtn2->Hide();

Label6->Hide();

Edit1->Hide();

choiceCurrencyCB->Hide();

Label7->Hide();

Label14->Hide();

Label10->Hide();

Label8->Hide();

Label9->Hide();

Label12->Hide();

Label13->Hide();

Edit2->Hide();

Label11->Hide();

}

}

}

void TForm1::AddToMemo(string line)

{

if (Memo1->Lines->Strings[0] == emptyCoupon.c\_str())

{

Memo1->Clear();

}

int pos = line.find('\n', 1);

string line1 = line.substr(0, pos);

Memo1->Lines->Add(line1.c\_str());

line = line.substr(pos + 1, line.length() - 1);

Memo1->Lines->Add(line.c\_str());

}

void TForm1::DeleteFromMemo(string line)

{

int pos = line.find('\n', 1);

int count = Memo1->Lines->Count;

for(int i = 0; i < count; i += 2)

{

string line1 = line.substr(0, pos);

if (Memo1->Lines->Strings[i] == line1.c\_str())

{

line1 = line.substr(pos + 1, line.length() - 2);//-2 ïîòîìó ÷òî â êîíöå åùå áóäåò çíàê òàáóëÿöèè

if (Memo1->Lines->Strings[i + 1] == line1.c\_str())

{

Memo1->Lines->Delete(i);

Memo1->Lines->Delete(i);

i -= 2;

}

}

}

if(Memo1->Lines->Count == 0)

{

Memo1->Clear();

Memo1->Lines->Add(emptyCoupon.c\_str());

}

}

void TForm1::MakeBet(TCheckListBox \*checkBox)

{

string dimension = (Label11->Caption).c\_str();

dimension = dimension.substr(1, dimension.size() - 1);

int dimens = StrToInt(dimension.c\_str());

int number = checkBox->ItemIndex;

string teamAndKoef = (checkBox->Items->operator [](number)).c\_str();

if (checkBox->Checked[number])

{

dimens++;

}

else

{

dimens--;

}

Label11->Caption = "/" + IntToStr(dimens);

if (dimens > 2)

{

string temp = (IntToStr(dimens - 1)).c\_str();

Edit2->Text = temp.c\_str();

}

else

{

Edit2->Text = "0";

}

switch(choiceBet->ItemIndex)

{

case 0:

AllInOneList(ordinarList, expressList, systemList);

ChoiceMatch(checkBox, ordinarList);

break;

case 1:

Label12->Show();

Label13->Show();

AllInOneList(expressList, ordinarList, systemList);

ChoiceMatch(checkBox, expressList);

break;

case 2:

Edit2->Show();

Label7->Show();

Label11->Show();

AllInOneList(systemList, expressList, ordinarList);

ChoiceMatch(checkBox, systemList);

if (dimens < 3)

{

BitBtn1->Hide();

BitBtn2->Hide();

}

break;

}

BetToForm();

}

void \_\_fastcall TForm1::homeCBClickCheck(TObject \*Sender)

{

MakeBet(homeCB);

Label15->Hide();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::drawCBClick(TObject \*Sender)

{

MakeBet(drawCB);

Label15->Hide();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::guestCBClick(TObject \*Sender)

{

MakeBet(guestCB);

Label15->Hide();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Edit1Change(TObject \*Sender)

{

if (isInitialize)

{

BetToForm();

}

}

//---------------------------------------------------------------------------

void TForm1::BetToForm()

{

int number = choiceCurrencyCB->ItemIndex;

string currency = (choiceCurrencyCB->Items->Strings[number]).c\_str();

try

{

float money = StrToFloat(Edit1->Text);

int dimension = StrToInt(Edit2->Text);

list <float> finalCoef = AllCoef(Memo1, choiceBet, money, dimension);

float allMoney = finalCoef.front();

finalCoef.pop\_front();

Label10->Caption = FloatToStr(allMoney) + " " + currency.c\_str();

float win = finalCoef.front();

finalCoef.pop\_front();

Label9->Caption = FloatToStr(win) + " " + currency.c\_str();

float maxKoef = finalCoef.front();

finalCoef.pop\_front();

Label13->Caption = FloatToStr(maxKoef);

}

catch (EConvertError &ex)

{

Label10->Caption = "0";

Label9->Caption = "0";

BitBtn1->Hide();

BitBtn2->Hide();

}

}

void \_\_fastcall TForm1::choiceBetChange(TObject \*Sender)

{

switch(choiceBet->ItemIndex)

{

case 0:

AllInOneList(ordinarList, expressList, systemList);

Label6->Caption = "Ñóììà íà êàæäûé îðäèíàð";

Edit2->Hide();

Label7->Hide();

Label10->Show();

Label11->Hide();

Label12->Hide();

Label13->Hide();

Label14->Show();

break;

case 1:

AllInOneList(expressList, ordinarList, systemList);

Label6->Caption = "Ñóììà íà âåñü ýêñïðåññ";

Edit2->Hide();

Label7->Hide();

Label10->Hide();

Label11->Hide();

Label14->Hide();

break;

case 2:

AllInOneList(systemList, expressList, ordinarList);

Label6->Caption = "Ñóììà íà ñèñòåìó";

string dimension = Label11->Caption.c\_str();

dimension = dimension.substr(1, dimension.length() - 1);

int dim = StrToInt(dimension.c\_str());

if (dim > 0)

{

Edit2->Show();

Label7->Show();

Label11->Show();

}

Label10->Hide();

Label12->Hide();

Label13->Hide();

Label14->Hide();

break;

}

BetToForm();

Label15->Hide();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Edit1Click(TObject \*Sender)

{

Edit1->Clear();

Label15->Hide();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::choiceCurrencyCBChange(TObject \*Sender)

{

BetToForm();

Label15->Hide();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn1Click(TObject \*Sender)

{

ClearForm();

}

void TForm1::ClearForm()

{

Memo1->Clear();

Memo1->Lines->Add(emptyCoupon.c\_str());

AllUnChecked(homeCB);

AllUnChecked(drawCB);

AllUnChecked(guestCB);

ordinarList.clear();

expressList.clear();

systemList.clear();

Edit1->Text = "0";

Label15->Hide();

Label6->Hide();

Edit1->Hide();

Edit2->Hide();

choiceCurrencyCB->Hide();

Label7->Hide();

Label14->Hide();

Label10->Hide();

Label8->Hide();

Label9->Hide();

Label11->Hide();

BitBtn1->Hide();

BitBtn2->Hide();

BetToForm();

}

//---------------------------------------------------------------------------

void TForm1::AllUnChecked(TCheckListBox \*checkBox)

{

int size = checkBox->Items->Count;

for (int i = 0; i < size; i++)

{

checkBox->Checked[i] = false;

}

}

void \_\_fastcall TForm1::BitBtn2Click(TObject \*Sender)

{

string path = Form4->userDirectory + Form4->user.login + "Bet.txt";

double moneyBet = StrToFloat(Edit1->Text);

double minbet = minBet();

int number = choiceCurrencyCB->ItemIndex;

string currency = (choiceCurrencyCB->Items->operator [](number)).c\_str();

Label12->Hide();

Label13->Hide();

if (moneyBet < minbet)

{

Label18->Caption = "Ìèíèìàëüíàÿ ñòàâêà " + FloatToStr(minbet) + " " + currency.c\_str();

Label18->Show();

}

else if(!isEnoughMoney(moneyBet))

{

Label18->Caption = "Íåäîñòàòî÷íî ñðåäñòâ";

Label18->Show();

}

else

{

SaveBetsToFile(path);

Label18->Hide();

}

}

//---------------------------------------------------------------------------

bool TForm1::isEnoughMoney(double moneyBet)

{

int number = choiceCurrencyCB->ItemIndex;

list<Cash> copyAllMoney = allMoney;

Cash cash;

for (int i = 0; i <= number; i++)

{

cash = copyAllMoney.front();

copyAllMoney.pop\_front();

}

float moneyInPurse = cash.money;

bool isEnough;

if (moneyBet <= moneyInPurse)

{

isEnough = true;

}

else

{

isEnough = false;

}

return isEnough;

}

double TForm1::minBet()

{

double minBet = 5000; //â áåë. ðóáëÿõ

switch (choiceCurrencyCB->ItemIndex)

{

case 1:

minBet /= 282;

minBet = int(minBet\*100 + 0.5)/100.0;

break;

case 2:

minBet /= 9300;

minBet = int(minBet\*100 + 0.5)/100.0;

break;

case 3:

minBet /= 12540;

minBet = int(minBet\*100 + 0.5)/100.0;

break;

}

return minBet;

}

void TForm1::SaveBetsToFile(string path)

{

try

{

ofstream file(path.c\_str(), ios::app);//ñîõðàíÿåì â ôàéë ñ äîáàâëåíèåì íîâûõ ñòàâîê

float money = StrToFloat(Edit1->Text);

int index = choiceCurrencyCB->ItemIndex;

string currency = (choiceCurrencyCB->Items->operator [](index)).c\_str();

switch(choiceBet->ItemIndex)

{

case 0:

SaveOrdinarToFile(ordinarList, file, currency, money);

ordinarList.clear();

break;

case 1:

SaveExpressToFile(expressList, file, currency, money);

expressList.clear();

break;

case 2:

string dimension = Edit2->Text.c\_str();

dimension += Label11->Caption.c\_str();

SaveSystemToFile(systemList, file, currency, money, dimension);

break;

}

ClearForm();

Label15->Show();

}

catch (EConvertError &ex)

{

Label10->Caption = "0";

Label9->Caption = "0";

Edit1->Text = "0";

}

}

void \_\_fastcall TForm1::N2Click(TObject \*Sender)//âûâîäèò äåíüãè íà ôîðìó

{

Form2->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N1Click(TObject \*Sender)

{

Form3->Show();

}

//---------------------------------------------------------------------------

void TForm1::MyBetToForm(ifstream& file, ofstream& fileCopy)

{

Form3->Memo1->Clear();

Form3->Memo2->Clear();

Form3->Memo3->Clear();

string currencyData = DateToStr(Form1->DateTimePicker1->Date).c\_str();

list<string> winMoney = Form3->LoadBet(file, fileCopy, currencyData);

Form2->WinMoney(winMoney);

}

void TForm1::MyMoneyToForm()

{

Form2->ToForm();

string path = Form4->path + Form4->user.login + "\\" + Form4->user.login + "Money.txt";

Form2->SaveMoneyToFile(path);

}

void \_\_fastcall TForm1::Edit2Change(TObject \*Sender)

{

try

{

int dim1 = StrToInt(Edit2->Text);

string dimension = Label11->Caption.c\_str();

dimension = dimension.substr(1, dimension.length());

int dim2 = StrToInt(dimension.c\_str());

if (dim1 >= dim2)

{

Edit2->Text = "0";

}

if (isInitialize)

{

BetToForm();

}

}

catch (EConvertError &ex)

{

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Edit2Click(TObject \*Sender)

{

Edit2->Clear();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N4Click(TObject \*Sender)

{

Form1->Hide();

Form5->Show();

}

//---------------------------------------------------------------------------

void TForm1::NotEnter()

{

Form1->GroupBox3->Hide();

Form1->Show();

Form1->N1->Visible = false;

Form1->N2->Visible = false;

Form1->N4->Visible = true;

Form1->N5->Visible = false;

Form1->userName1->Visible = false;

Form1->Label16->Show();

Form1->Label17->Show();

string pathRunOptions = "D:\\Ó÷åáà\\Ïðîãðàììèðîâàíèå\\Êóðñà÷\\SportsBetting\\RunOptions.txt";

ofstream fileRunOptions(pathRunOptions.c\_str());//î÷èùàåì ôàéë

allMoney.clear();

}

void TForm1::Enter()

{

Form1->Show();

Form1->N1->Visible = true;

Form1->N2->Visible = true;

Form1->N4->Visible = false;

Form1->N5->Visible = false;

Form1->userName1->Visible = true;

Form1->userName1->Caption = Form4->user.login.c\_str();

Form1->Label16->Hide();

Form1->Label17->Hide();

Form1->GroupBox3->Show();

if (Form1->userName1->Caption == "&administrator")

{

Form1->N5->Visible = true;

}

}

void \_\_fastcall TForm1::N3Click(TObject \*Sender)

{

Form1->NotEnter();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N5Click(TObject \*Sender)

{

Form6->OpenStatistic();

Form6->DateTimePicker1Change(Sender); //âûçûâàåì îáðàáîò÷èê ñîáûòèÿ DateTimePicker1Change

//êîòîðûé âûçûâàåò CalculateStatistic

Form6->Show();

}

//---------------------------------------------------------------------------

#ifndef Unit2H

#define Unit2H

//---------------------------------------------------------------------------

class TForm2 : public TForm

{

\_\_published: // IDE-managed Components

TButton \*Button1;

TButton \*Button2;

TCheckListBox \*CheckListBox1;

TValueListEditor \*ValueListEditor1;

TBitBtn \*BitBtn1;

void \_\_fastcall CheckListBox1ClickCheck(TObject \*Sender);

void \_\_fastcall Button1Click(TObject \*Sender);

void \_\_fastcall BitBtn1Click(TObject \*Sender);

void \_\_fastcall Button2Click(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TForm2(TComponent\* Owner);

void ToForm();

void ChangeMoney();

void WinMoney(list<string> winMoney);

void SaveMoneyToFile(string pathFile);

bool isAdd;

};

//---------------------------------------------------------------------------

extern PACKAGE TForm2 \*Form2;

//---------------------------------------------------------------------------

#endif

void TForm2::SaveMoneyToFile(string pathFile)

{

ofstream file(pathFile.c\_str());

list<Cash> copyAllMoney = allMoney;

int size = copyAllMoney.size();

for (int i = 0; i < size; i++)

{

Cash temp = copyAllMoney.front();

copyAllMoney.pop\_front();

file << (FloatToStr(temp.money)).c\_str() << " ";

file << temp.currency << "\n";

}

}

void \_\_fastcall TForm2::CheckListBox1ClickCheck(TObject \*Sender)

{

int number = CheckListBox1->ItemIndex;

if (CheckListBox1->Checked[number])

{

Button1->Enabled = true;

Button2->Enabled = true;

}

else

{

Button1->Enabled = false;

Button2->Enabled = false;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm2::Button1Click(TObject \*Sender)

{

ValueListEditor1->Show();

Button1->Hide();

Button2->Hide();

int size = CheckListBox1->Items->Count;

int j = 1;

for(int i = 0; i < size; i++)

{

if (CheckListBox1->Checked[i])

{

ValueListEditor1->Keys[j] = CheckListBox1->Items->operator [](i);

ValueListEditor1->Strings->Add("");

j++;

}

}

ValueListEditor1->Strings->Delete(j-1);

BitBtn1->Show();

BitBtn1->Caption = "Âûâåñòè äåíüãè";

isAdd = false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm2::BitBtn1Click(TObject \*Sender)

{

ChangeMoney();

}

//---------------------------------------------------------------------------

void TForm2::ToForm()

{

CheckListBox1->Clear();

Button1->Show();

Button1->Enabled = false;

Button2->Show();

Button2->Enabled = false;

BitBtn1->Hide();

ValueListEditor1->Hide();

list<Cash> copyAllMoney = allMoney;

int size = copyAllMoney.size();

for (int i = 0; i < size; i++)

{

Cash temp = copyAllMoney.front();

copyAllMoney.pop\_front();

string money = (FloatToStr(temp.money)).c\_str();

string toListbox = money + " " + temp.currency;

Form2->CheckListBox1->Items->Add(toListbox.c\_str());

}

}

void TForm2::ChangeMoney()

{

int size = CheckListBox1->Items->Count;

int j = 1;

list<Cash> copyAllMoney = allMoney;

allMoney.clear();

for(int i = 0; i < size; i++)

{

Cash newCash = copyAllMoney.front();

copyAllMoney.pop\_front();

string temp = (CheckListBox1->Items->operator [](i)).c\_str();

int pos = temp.find(' ', 1);

temp = temp.substr(pos + 1, temp.length() - 1);

string temp1 = ValueListEditor1->Keys[j].c\_str();

if ((CheckListBox1->Items->operator [](i) == ValueListEditor1->Keys[j]) || (temp == temp1))

{

string temp = (ValueListEditor1->Strings->operator [](j-1)).c\_str();

int position = temp.find('=',1);

temp = temp.substr(position + 1, temp.length() - 1);

if (isAdd)

{

newCash.money = newCash.money + StrToFloat(temp.c\_str());

}

else

{

newCash.money = newCash.money - StrToFloat(temp.c\_str());

}

j++;

}

if (j > (ValueListEditor1->RowCount - 1))

j--;

allMoney.push\_back(newCash);

}

CheckListBox1->Clear();

ToForm();//ìåíÿåì çíà÷åíèÿ â ñ÷åòå

SaveMoneyToFile(Form4->user.login);

}

void TForm2::WinMoney(list<string> winMoney)

{

int size = winMoney.size();

for (int i = 0; i < size; i++)

{

string win = winMoney.front();

winMoney.pop\_front();

int pos = win.find(' ', 1);

float money = StrToFloat((win.substr(0, pos)).c\_str());

string currency = win.substr(pos + 1, win.length() - 1);

int sizeAllMoney = allMoney.size();

list<Cash> copyAllMoney = allMoney;

allMoney.clear();

for(int j = 0; j < sizeAllMoney; j++)

{

Cash cash = copyAllMoney.front();

copyAllMoney.pop\_front();

if (cash.currency == currency)

{

cash.money += money;

}

allMoney.push\_back(cash);

}

}

}

void \_\_fastcall TForm2::Button2Click(TObject \*Sender)

{

ValueListEditor1->Show();

Button1->Hide();

Button2->Hide();

int size = CheckListBox1->Items->Count;

int j = 1;

for(int i = 0; i < size; i++)

{

if (CheckListBox1->Checked[i])

{

string temp = (CheckListBox1->Items->operator [](i)).c\_str();

int pos = temp.find(' ', 1);

temp = temp.substr(pos + 1, temp.length() - 1);

ValueListEditor1->Keys[j] = temp.c\_str();

ValueListEditor1->Strings->Add("");

j++;

}

}

ValueListEditor1->Strings->Delete(j-1);

BitBtn1->Show();

BitBtn1->Caption = "Ââåñòè äåíüãè";

isAdd = true;

}

//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

#ifndef Unit3H

#define Unit3H

//---------------------------------------------------------------------------

class TForm3 : public TForm

{

\_\_published: // IDE-managed Components

TPageControl \*PageControl1;

TTabSheet \*TabSheet1;

TTabSheet \*TabSheet2;

TMemo \*Memo1;

TMemo \*Memo2;

TTabSheet \*TabSheet3;

TMemo \*Memo3;

void \_\_fastcall FormCreate(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TForm3(TComponent\* Owner);

list<string> LoadBet(ifstream& file, ofstream& filecopy, string currencyData);

string LoadOrdinarMBet(ifstream& file, ofstream& fileCopy, string currencyData);

string LoadExpressMBet(ifstream& file, ofstream& fileCopy, string currencyData);

string LoadSystemMBet(ifstream& file, ofstream& fileCopy, string currencyData);

Bet LoadOneBet(ifstream& file);

};

//---------------------------------------------------------------------------

extern PACKAGE TForm3 \*Form3;

//---------------------------------------------------------------------------

#endif

//---------------------------------------------------------------------------

#include <vcl.h>

#include <iostream>

#include <string>

#include <Windows.h>

#include <string.h>

#include <fstream>

#include <locale.h>

#include <list>

#include <cstring.h>

#pragma hdrstop

#include "Unit3.h"

#include "Unit1.h"

#include "MadeBet.cpp"

using namespace std;

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm3 \*Form3;

//---------------------------------------------------------------------------

\_\_fastcall TForm3::TForm3(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm3::FormCreate(TObject \*Sender)

{

Memo1->Clear();

Memo2->Clear();

Memo3->Clear();

LoadResult();

}

//---------------------------------------------------------------------------

list<string> TForm3::LoadBet(ifstream& file, ofstream& fileCopy, string currencyData)

{

string str;

list<string> winMoneyList;

while(!file.eof())

{

string winAndCurrency = "";

getline(file, str, '\t');

string temp; //îáúÿâèëè ïåðåìåííóþ temp ò.ê. â ñëåäóþùåé ñòðî÷êå

getline(file, temp, '\n');//str èçìåíèëñÿ áû. à îí íàì äàëüøå íóæåí

if(str == "Îðäèíàð")

{

winAndCurrency = LoadOrdinarMBet(file, fileCopy, currencyData);

}

else if ((str == "Ýêñïðåññ") || (str == "\nÝêñïðåññ"))

{

winAndCurrency = LoadExpressMBet(file, fileCopy, currencyData);

}

else if ((str == "Ñèñòåìà") || (str == "\nÑèñòåìà"))

{

winAndCurrency = LoadSystemMBet(file, fileCopy, currencyData);

}

if (winAndCurrency != "")

{

winMoneyList.push\_back(winAndCurrency);

}

}

return winMoneyList;

}

Bet TForm3::LoadOneBet(ifstream& file)

{

Bet newBet;

string str;

getline(file, str, '\t');

if ((str == "/Ýêñïðåññ") || (str == "/Ýêñïðåññ\n") || (str == "/Ñèñòåìà") || (str == "/Ñèñòåìà\n"))

{

return newBet;//âîçâðàòèòñÿ çíà÷åíèå ïåðåìåííîé òèïà Bet ïî óìîë÷àíèþ

}

newBet.date = str;

getline(file, str, ' ');

newBet.home = str;

getline(file, str, ' ');

newBet.guest = str;

getline(file, str, ' ');

newBet.resultMatch = str;

getline(file, str, ' ');

newBet.coefficient = StrToFloat(str.c\_str());

getline(file, str, ' ');

newBet.money = StrToFloat(str.c\_str());

getline(file, str, ' ');

newBet.currency = str;

getline(file, str, ' ');

newBet.isBetTreatment = StrToBool(str.c\_str());

getline(file, str, '\n');

newBet.winning = StrToFloat(str.c\_str());

return newBet;

}

string TForm3::LoadOrdinarMBet(ifstream& file, ofstream& fileCopy, string currencyData)

{

Bet newBet = LoadOneBet(file);

float win = 0;//âûèãðàííûå äåíüãè

string winAndCurrency = "";//åñëè òàêîå æå çíà÷åíèå áóäåò âîçâðàùåíî ôóíêöèåé, òî çíà÷èò âûèãðûøà íåò

if(newBet.date <= currencyData)

{

int size = mBResult.size();

list <MResult> copymBResult = mBResult;

for (int i = 0; i < size; i++)

{

MResult tempResult = copymBResult.front();

copymBResult.pop\_front();

if ((tempResult.home == newBet.home) && (tempResult.guest == newBet.guest))

{

if (tempResult.date == newBet.date)

{

//Ñòàâêà ñûãðàëà è íå ðàññ÷èòàíà

if((tempResult.result == newBet.resultMatch) && (newBet.isBetTreatment == false))

{

ToMemo(newBet, Memo2);

win = newBet.money \* newBet.coefficient;

Memo2->Lines->Add("\tÂûèãðûø: " + FloatToStr(win));

string win1 = (FloatToStr(win)).c\_str();

winAndCurrency = win1 + " " + newBet.currency;

newBet.isBetTreatment = true;

newBet.winning = win;

}

//ñòàâêà ñûãðàëà è ðàññ÷èòàíà

else if((tempResult.result == newBet.resultMatch) && (newBet.isBetTreatment == true))

{

ToMemo(newBet, Memo2);

win = newBet.money \* newBet.coefficient;

Memo2->Lines->Add("\tÂûèãðûø: " + FloatToStr(win));

}

//ñòàâêà íå ñûãðàëà

else

{

ToMemo(newBet, Memo3);

Memo3->Lines->Add("\tÂûèãðûø: 0");

newBet.isBetTreatment = true;

newBet.winning = -newBet.money;

}

}

}

}

}

else

{

ToMemo(newBet, Memo1);

}

List temp;

temp.push\_back(newBet);

SaveOrdinarToFile(temp, fileCopy, defaultCurrency, defaultMoney);

temp.clear();

return winAndCurrency;

}

//Çàãðóæàåò òîëüêî îäèí ýêñïðåññ

string TForm3::LoadExpressMBet(ifstream& file, ofstream& fileCopy, string currencyData)

{

string str;

bool isWin = false;

bool isLose = false;

bool isCurrent = false;

bool isGotMoney = false;

List temp;

float win = 0;//âûèãðàííûå äåíüãè

float money;

string currency;

float coefExpress = 1;

string winAndCurrency = "";//åñëè òàêîå æå çíà÷åíèå áóäåò âîçâðàùåíî ôóíêöèåé, òî çíà÷èò âûèãðûøà íåò

string forMemo = "Ýêñïðåññ\n";

Bet newBet;

while(!file.eof())

{

Bet newBet = LoadOneBet(file);

if (!newBet.date.empty())

{

if(newBet.date <= currencyData)

{

money = newBet.money;

currency = newBet.currency;

int size = mBResult.size();

list <MResult> copymBResult = mBResult;

for (int i = 0; i < size; i++)

{

MResult tempResult = copymBResult.front();

copymBResult.pop\_front();

if ((tempResult.home == newBet.home) && (tempResult.guest == newBet.guest))

{

if (tempResult.date == newBet.date)

{

//Ñòàâêà ñûãðàëà è íå ðàññ÷èòàíà

if((tempResult.result == newBet.resultMatch) && (newBet.isBetTreatment == false))

{

coefExpress \*= newBet.coefficient;

newBet.isBetTreatment = true;

isWin = true;

isGotMoney = false;

}

else if((tempResult.result == newBet.resultMatch) && (newBet.isBetTreatment == true))

{

coefExpress \*= newBet.coefficient;

isWin = true;

isGotMoney = true;

}

//Ñòàâêà íå ñûãðàëà

else

{

newBet.isBetTreatment = true;

isLose = true;

}

}

else

{

isCurrent = true;

}

string kef = FloatToStr(newBet.coefficient).c\_str();

forMemo += newBet.date + " " + newBet.home + " " + newBet.guest + " " + newBet.resultMatch + " Êô: " + kef + "\n";

temp.push\_back(newBet);

}

}

}

else

{

isCurrent = true;

string kef = FloatToStr(newBet.coefficient).c\_str();

forMemo += newBet.date + " " + newBet.home + " " + newBet.guest + " " + newBet.resultMatch + " Êô: " + kef + "\n";

temp.push\_back(newBet);

}

}

else

{

getline(file, str, '\n');

break;

}

}//äëÿ while

string koef1 = (FloatToStr(coefExpress)).c\_str();

string money1 = (FloatToStr(newBet.money)).c\_str();

forMemo += "Êô: " + koef1 + " Ñóììà: " + money1 + " " + newBet.currency + "\n";

List copyTemp = temp;

temp.clear();

int size = copyTemp.size();

if (isCurrent)//òåêöùèé ýêñïðåññ

{

win = coefExpress \* money;

ExpressAndSystemToMemo(forMemo, Memo1);

for (int i = 0; i < size; i++)

{

Bet bet = copyTemp.front();

copyTemp.pop\_front();

bet.isBetTreatment = false;

bet.winning = 0;

temp.push\_back(bet);

}

}

else if (isWin && !isLose && !isCurrent) //ýêñïðåññ ñûãðàë

{

win = coefExpress \* money;

string win1 = (FloatToStr(win)).c\_str();

forMemo += "\tÂûèãðûø: " + win1 + "\n";

if (!isGotMoney)

{

winAndCurrency = win1 + " " + currency;

}

ExpressAndSystemToMemo(forMemo, Memo2);

for (int i = 0; i < size; i++)

{

Bet bet = copyTemp.front();

copyTemp.pop\_front();

bet.winning = win;

temp.push\_back(bet);

}

}

else if (isLose)

{

forMemo += "\tÂûèãðûø: 0\n";

ExpressAndSystemToMemo(forMemo, Memo3);

for (int i = 0; i < size; i++)

{

Bet bet = copyTemp.front();

copyTemp.pop\_front();

bet.isBetTreatment = true;

bet.winning = 0;

temp.push\_back(bet);

}

}

SaveExpressToFile(temp, fileCopy, defaultCurrency, defaultMoney);

temp.clear();

return winAndCurrency;

}

void ExpressAndSystemToMemo(string forMemo, TMemo \*Memo)

{

while(!forMemo.empty())

{

int pos = forMemo.find('\n', 1);

string temp = forMemo.substr(0, pos);

forMemo = forMemo.substr(pos + 1, forMemo.length());

Memo->Lines->Add(temp.c\_str());

}

}

string TForm3::LoadSystemMBet(ifstream& file, ofstream& fileCopy, string currencyData)

{

string str;

bool isWin = false;

bool isLose = false;

bool isCurrent = false;

bool isGotMoney = false;

List temp;

float win = 0;//âûèãðàííûå äåíüãè

float money;

string currency;

string winAndCurrency = "";//åñëè òàêîå æå çíà÷åíèå áóäåò âîçâðàùåíî ôóíêöèåé, òî çíà÷èò âûèãðûøà íåò

Bet newBet;

getline(file, str, '\n');

string dimension = str;

int dim1 = StrToInt(dimension[0]);

int dim2 = StrToInt(dimension[2]);

int canLostGames = dim2 - dim1;

int lostGames = 0;

list<float> coefList;

string forMemo = "Ñèñòåìà\n";

forMemo += "Ðàçìåðíîñòü: " + dimension + "\n";

while(!file.eof())

{

Bet newBet = LoadOneBet(file);

if (!newBet.date.empty())

{

if(newBet.date <= currencyData)

{

money = newBet.money;

currency = newBet.currency;

int size = mBResult.size();

list <MResult> copymBResult = mBResult;

for (int i = 0; i < size; i++)

{

MResult tempResult = copymBResult.front();

copymBResult.pop\_front();

if ((tempResult.home == newBet.home) && (tempResult.guest == newBet.guest))

{

if (tempResult.date == newBet.date)

{

//Ñòàâêà ñûãðàëà è íå ðàññ÷èòàíà

if((tempResult.result == newBet.resultMatch) && (newBet.isBetTreatment == false))

{

coefList.push\_back(newBet.coefficient);

newBet.isBetTreatment = true;

isWin = true;

isGotMoney = false;

}

//ñòàâêà ñûãðàëà è ðàññ÷èòàíà

else if((tempResult.result == newBet.resultMatch) && (newBet.isBetTreatment == true))

{

coefList.push\_back(newBet.coefficient);

isWin = true;

isGotMoney = true;

}

//Ñòàâêà íå ñûãðàëà

else

{

coefList.push\_back(0);

newBet.isBetTreatment = true;

if (lostGames >= canLostGames)

{

isLose = true;

}

lostGames++;

}

}

else

{

isCurrent = true;

}

forMemo += newBet.date + " " + newBet.home + " " + newBet.guest + " " + newBet.resultMatch + "\n";

temp.push\_back(newBet);

}

}

}

else

{

forMemo += newBet.date + " " + newBet.home + " " + newBet.guest + " " + newBet.resultMatch + "\n";

temp.push\_back(newBet);

}

}

else

{

getline(file, str, '\n');

break;

}

}//äëÿ while

string money1 = (FloatToStr(money)).c\_str();

List copyTemp = temp;

temp.clear();

int size = copyTemp.size();

if (isCurrent) //òåêóùàÿÿ ñòàâêà

{

forMemo += "Ðàçìåðíîñòü: " + dimension + " Ñóììà: " + money1 + " " + newBet.currency + "\n";

ExpressAndSystemToMemo(forMemo, Memo1);

for (int i = 0; i < size; i++)

{

Bet bet = copyTemp.front();

copyTemp.pop\_front();

bet.isBetTreatment = false;

bet.winning = 0;

temp.push\_back(bet);

}

}

if (isWin && !isLose && !isCurrent) //ñòàâêà ñûãðàëà

{

list <float> temp1 = CalculateSystem(coefList, money, dim1);

temp1.pop\_front();

win = temp1.front();

forMemo += " Ñóììà: " + money1 + " " + newBet.currency + "\n";

string winnn = (FloatToStr(win)).c\_str();

forMemo += "\tÂûèãðûø: " + winnn + "\n";

if (!isGotMoney)

{

string win1 = (FloatToStr(win)).c\_str();

winAndCurrency = win1 + " " + currency;

}

ExpressAndSystemToMemo(forMemo, Memo2);

for (int i = 0; i < size; i++)

{

Bet bet = copyTemp.front();

copyTemp.pop\_front();

bet.winning = win;

temp.push\_back(bet);

}

}

else if (isLose)//ñòàâêà íå ñûãðàëà

{

forMemo += "Ðàçìåðíîñòü: " + dimension + " Ñóììà: " + money1 + " " + newBet.currency + "\n";

forMemo += "\tÂûèãðûø: 0\n";

ExpressAndSystemToMemo(forMemo, Memo3);

for (int i = 0; i < size; i++)

{

Bet bet = copyTemp.front();

copyTemp.pop\_front();

bet.isBetTreatment = true;

bet.winning = -bet.money;

temp.push\_back(bet);

}

}

SaveSystemToFile(temp, fileCopy, defaultCurrency, defaultMoney, dimension);

temp.clear();

return winAndCurrency;

}

void ToMemo(Bet bet, TMemo \*Memo)

{

string lines = bet.date + " " + bet.home + " " + bet.guest + " " + bet.resultMatch ;

string koef = (FloatToStr(bet.coefficient)).c\_str();

string money = (FloatToStr(bet.money)).c\_str();

lines += " Êô: " + koef + " Ñóììà: " + money + " " + bet.currency;

Memo->Lines->Add("Îðäèíàð");

Memo->Lines->Add(lines.c\_str());

}

//---------------------------------------------------------------------------

#ifndef Unit4H

#define Unit4H

//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

#include <vcl.h>

#include <iostream>

#include <string>

#include <Windows.h>

#include <string.h>

#include <fstream>

#include <locale.h>

#include <list>

#include <cstring.h>

#pragma hdrstop

#include "Unit4.h"

#include "Unit2.h"

#include "Money.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm4 \*Form4;

//---------------------------------------------------------------------------

\_\_fastcall TForm4::TForm4(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::FormCreate(TObject \*Sender)

{

//BitBtn1->Hide();

path = "D:\\Ó÷åáà\\Ïðîãðàììèðîâàíèå\\Êóðñà÷\\SportsBetting\\Users\\";

Edit1->Clear();

Edit2->Clear();

Edit3->Clear();

Edit4->Clear();

Edit5->Clear();

Memo1->Clear();

string webMoneyPurse[4] = {"B", "R", "Z", "E"};

for (int i = 1; i <= 4; i++)

{

webMoneyCB->Items->Add((webMoneyPurse[i-1]).c\_str());

}

webMoneyCB->ItemIndex = 0;

for (int i = 1; i <= 31; i++)

{

dateCB->Items->Add(IntToStr(i));

}

dateCB->ItemIndex = 0;

string months[12] = {"ßíâàðü", "Ôåâðàëü", "Ìàðò"

, "Àïðåëü", "Ìàé", "Èþíü"

, "Èþëü", "Àâãóñò", "Ñåíòÿáðÿ"

, "Îêòÿáðü", "Íîÿáðü", "Äåêàáðü"};

for (int i = 1; i <= 12; i++)

{

monthCB->Items->Add((months[i-1]).c\_str());

}

monthCB->ItemIndex = 0;

for (int i = 1995; i >= 1920; i--)

{

yearsCB->Items->Add(IntToStr(i));

}

yearsCB->ItemIndex = 0;

}

//---------------------------------------------------------------------------

void TForm4::AddUser()

{

user.login = Edit1->Text.c\_str();

user.password = Edit2->Text.c\_str();

user.mail = Edit4->Text.c\_str();

int number = webMoneyCB->ItemIndex;

string currency = (webMoneyCB->Items->operator [](number)).c\_str();

string numberPurse = Edit5->Text.c\_str();

user.webMoneyPurse = currency + numberPurse;

if (RadioButton1->Checked)

{

user.gender = RadioButton1->Caption.c\_str();

}

else

{

user.gender = RadioButton2->Caption.c\_str();

}

number = dateCB->ItemIndex;

string date = (dateCB->Items->operator [](number)).c\_str();

number = monthCB->ItemIndex;

string month = (monthCB->Items->operator [](number)).c\_str();

number = yearsCB->ItemIndex;

string years = (yearsCB->Items->operator [](number)).c\_str();

user.dateOfBirth = date + "." + month + "." + years;

number = Memo1->Lines->Count;

for (int i = 0; i < number; i++)

{

user.location += (Memo1->Lines->operator [](i)).c\_str();

user.location += " ";

}

}

void TForm4::SaveUserToFile(string pathFile)

{

ofstream file(pathFile.c\_str());//ôàéë î÷èùàåòñÿ

file << "Ïîëüçîâàòåëü" << "\n";

file << user.login << " ";

file << user.password << " ";

file << user.mail << " ";

file << user.webMoneyPurse << " ";

file << user.gender << " ";

file << user.dateOfBirth << " ";

file << user.location << "\n";

file.close();

}

void \_\_fastcall TForm4::BitBtn1Click(TObject \*Sender)

{

AddUser();

string login = user.login;

userDirectory = path + login + "\\";

CreateDirectory((userDirectory).c\_str(), NULL);

string pathFile = userDirectory + login + ".txt";

SaveUserToFile(pathFile);

MoneyDefault();

pathFile = userDirectory + login + "Money.txt";

Form2->SaveMoneyToFile(pathFile);

ofstream fileBet((Form4->userDirectory + login + "Bet.txt").c\_str());//ïðîñòî ñîçäàåì ôàéë

fileBet.close();

}

//---------------------------------------------------------------------------

class TForm5 : public TForm

{

\_\_published: // IDE-managed Components

TLabel \*Label1;

TLabel \*Label2;

TEdit \*Edit1;

TEdit \*Edit2;

TBitBtn \*BitBtn1;

TButton \*Button1;

TLabel \*Label3;

TCheckBox \*CheckBox1;

void \_\_fastcall FormCreate(TObject \*Sender);

void \_\_fastcall Button1Click(TObject \*Sender);

void \_\_fastcall BitBtn1Click(TObject \*Sender);

void \_\_fastcall FormClose(TObject \*Sender, TCloseAction &Action);

void \_\_fastcall Edit1Change(TObject \*Sender);

void \_\_fastcall Edit2Change(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TForm5(TComponent\* Owner);

void LoadUser(ifstream& file);

void LoadAll(string login);

list<string> FindAllUsers();

string GetPassword(string login);

bool Enter(string login, string password);

void RememberMe(string login, string password);

bool EnterInProfile();

};

//---------------------------------------------------------------------------

extern PACKAGE TForm5 \*Form5;

//---------------------------------------------------------------------------

#endif

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit5.h"

#include "Unit4.h"

#include "Unit3.h"

#include "Unit1.h"

#include "Money.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm5 \*Form5;

//---------------------------------------------------------------------------

\_\_fastcall TForm5::TForm5(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm5::FormCreate(TObject \*Sender)

{

Edit1->Clear();

Edit2->Clear();

Label3->Hide();

if(EnterInProfile())

{

Form1->Enter();

}

else

{

Form5->Show();

}

}

//---------------------------------------------------------------------------

bool TForm5::EnterInProfile()

{

string pathFile = "D:\\Ó÷åáà\\Ïðîãðàììèðîâàíèå\\Êóðñà÷\\SportsBetting\\RunOptions.txt";

ifstream file(pathFile.c\_str());

string login, password;

bool isEnter = false;

while(!file.eof())

{

string temp;

getline(file, temp, ' ');

if ((!temp.empty()) && (temp != "\n"))

{

login = temp;

getline(file, temp, '\n');

password = temp;

isEnter = true;

Form5->Enter(login, password);

break;

}

}

file.close();

return isEnter;

}

void \_\_fastcall TForm5::Button1Click(TObject \*Sender)

{

Form5->Hide();

Form4->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm5::BitBtn1Click(TObject \*Sender)

{

string login = Edit1->Text.c\_str();

string password = Edit2->Text.c\_str();

if(Enter(login, password))

{

RememberMe(login, password);

}

}

void TForm5::RememberMe(string login, string password)

{

string pathRunOptions = "D:\\Ó÷åáà\\Ïðîãðàììèðîâàíèå\\Êóðñà÷\\SportsBetting\\RunOptions.txt";

if (CheckBox1->Checked)

{

ofstream fileRunOptions(pathRunOptions.c\_str(), ios::app);

fileRunOptions << login << " ";

fileRunOptions << password << "\n";

fileRunOptions.close();

}

else

{

ofstream fileRunOptions(pathRunOptions.c\_str());//ñîäåðæèìîå ôàéëà î÷èùàåòñÿ

fileRunOptions.close();

}

}

bool TForm5::Enter(string login, string password)

{

list<string> allUsers = Form5->FindAllUsers();

int size = allUsers.size();

bool isEnter = false;

for (int i = 0; i < size; i++)

{

string directoryName = allUsers.front();

allUsers.pop\_front();

Form4->userDirectory = Form4->path + directoryName + "\\";

if(directoryName == login)

{

if(password == GetPassword(login))

{

Form5->Hide();

LoadAll(login);

isEnter = true;

break;

}

}

}

if (isEnter)

{

Form1->Enter();

}

else

{

Label3->Show();

}

return isEnter;

}

string TForm5::GetPassword(string login)

{

string pathFile = Form4->userDirectory + login + ".txt";

ifstream file(pathFile.c\_str());

string password = "";

while(!file.eof())

{

string temp;

getline(file, temp, '\n');

if ((!temp.empty()) && (temp != "\n") && (temp == "Ïîëüçîâàòåëü"))

{

getline(file, temp, ' ');

if (temp == login)

{

getline(file, password, ' ');

break;

}

}

}

file.close();

return password;

}

//---------------------------------------------------------------------------

void TForm5::LoadAll(string login)

{

string pathFile = Form4->userDirectory + login + ".txt";

ifstream file(pathFile.c\_str());

LoadUser(file);

string pathMoney = Form4->userDirectory + login + "Money.txt";

LoadMoneyFromFile(pathMoney);

string pathBet = Form4->userDirectory + login + "Bet.txt";

string pathBetCopy = Form4->userDirectory + "copy" + login + "Bet.txt";

ifstream fileBet(pathBet.c\_str());

ofstream fileBetCopy(pathBetCopy.c\_str());

Form1->MyBetToForm(fileBet, fileBetCopy);

Form1->MyMoneyToForm();

/////////////////////////

fileBet.close();

fileBetCopy.close();

ifstream ifs(pathBetCopy.c\_str()); //Ïîòîì ïåðåìåñòèòü êóäà-íèáóääü

ofstream ofs(pathBet.c\_str());

string stri;

while (getline(ifs,stri))

ofs << stri << '\n';

///////////////////////////////

}

void TForm5::LoadUser(ifstream& file)

{

string str;

while(!file.eof())

{

getline(file, str, '\n');

if ((!str.empty()) && (str != "\n") && (str == "Ïîëüçîâàòåëü"))

{

User user = Form4->user;

getline(file, str, ' ');

user.login = str;

getline(file, str, ' ');

user.password = str;

getline(file, str, ' ');

user.mail = str;

getline(file, str, ' ');

user.webMoneyPurse = str;

getline(file, str, ' ');

user.gender = str;

getline(file, str, ' ');

user.dateOfBirth = str;

getline(file, str, '\n');

user.location = str;

Form4->user = user;

break;

}

}

}

list<string> TForm5::FindAllUsers()

{

string folder\_path = Form4->path.c\_str();

folder\_path += "\*";

HANDLE hSearch;

WIN32\_FIND\_DATA pFileData;

hSearch = FindFirstFile(folder\_path.c\_str(), &pFileData);

list<string> allUsers;

if (hSearch != INVALID\_HANDLE\_VALUE)

{

do

{

if ((pFileData.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY))

{

string directoryName = pFileData.cFileName;

if ((directoryName != ".") && (directoryName != ".."))

{

allUsers.push\_back(directoryName);

}

}

}

while (FindNextFile(hSearch, &pFileData));

}

FindClose(hSearch);

return allUsers;

}

void \_\_fastcall TForm5::FormClose(TObject \*Sender, TCloseAction &Action)

{

Form1->NotEnter();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm5::Edit1Change(TObject \*Sender)

{

Label3->Hide();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm5::Edit2Change(TObject \*Sender)

{

Label3->Hide();

}

//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

#ifndef Unit6H

#define Unit6H

//---------------------------------------------------------------------------

class TForm6 : public TForm

{

\_\_published: // IDE-managed Components

TLabel \*Label1;

TLabel \*Label2;

TDateTimePicker \*DateTimePicker1;

TDateTimePicker \*DateTimePicker2;

TLabel \*Label3;

TLabel \*Label4;

TLabel \*Label5;

TLabel \*Label6;

TEdit \*Edit1;

TEdit \*Edit2;

TLabel \*Label7;

TEdit \*Edit3;

TLabel \*Label8;

TEdit \*Edit4;

TLabel \*Label9;

TLabel \*Label10;

void \_\_fastcall FormCreate(TObject \*Sender);

void \_\_fastcall DateTimePicker1Change(TObject \*Sender);

void \_\_fastcall DateTimePicker2Change(TObject \*Sender);

void \_\_fastcall Label4Click(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TForm6(TComponent\* Owner);

void OpenStatistic();

void CalculateStatistic(TDate date1, TDate date2);

};

//---------------------------------------------------------------------------

extern PACKAGE TForm6 \*Form6;

struct OneBet

{

string date;

string currency;

float win;

};

list<OneBet> allStatistic;

string officeBirthDay = "07.11.2013";

TDate dateOfficeBirthDay = StrToDate(officeBirthDay.c\_str());

//---------------------------------------------------------------------------

#endif

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit6.h"

#include "Unit5.h"

#include "Unit4.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm6 \*Form6;

//---------------------------------------------------------------------------

\_\_fastcall TForm6::TForm6(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void TForm6::OpenStatistic()

{

list<string> AllUsers = Form5->FindAllUsers();

int size = AllUsers.size();

for (int i = 0; i < size; i++)

{

string login = AllUsers.front();

AllUsers.pop\_front();

if(login == "administrator")

{

continue;

}

string pathBet = Form4->path + login + "\\" + login + "Bet.txt";

ifstream file(pathBet.c\_str());

string str;

OneBet bet;

while(!file.eof())

{

getline(file, str, '\t');

string temp; //îáúÿâèëè ïåðåìåííóþ temp ò.ê. â ñëåäóþùåé ñòðî÷êå

getline(file, temp, '\n');//str èçìåíèëñÿ áû. à îí íàì äàëüøå íóæåí

if (str.empty())

{

continue;

}

if(str == "Îðäèíàð")

{

getline(file, str, ' ');

bet.date = str;

getline(file, str, ' ');

getline(file, str, ' ');

getline(file, str, ' ');

getline(file, str, ' ');

getline(file, str, ' ');

getline(file, str, ' ');

bet.currency = str;

getline(file, str, ' ');

getline(file, str, '\n');

bet.win = StrToFloat(str.c\_str());

}

else if ((str == "Ýêñïðåññ") || (str == "\nÝêñïðåññ"))

{

getline(file, str, '\t');

bet.date = str;

getline(file, str, ' ');

getline(file, str, ' ');

getline(file, str, ' ');

getline(file, str, ' ');

getline(file, str, ' ');

getline(file, str, ' ');

bet.currency = str;

getline(file, str, ' ');

getline(file, str, '\n');

bet.win = StrToFloat(str.c\_str());

do

{

getline(file, str, '\n');

}while (str != "/Ýêñïðåññ\t");

}

else if ((str == "Ñèñòåìà") || (str == "\nÑèñòåìà"))

{

getline(file, str, '\n');

getline(file, str, '\t');

bet.date = str;

getline(file, str, ' ');

getline(file, str, ' ');

getline(file, str, ' ');

getline(file, str, ' ');

getline(file, str, ' ');

getline(file, str, ' ');

bet.currency = str;

getline(file, str, ' ');

getline(file, str, '\n');

bet.win = StrToFloat(str.c\_str());

do

{

getline(file, str, '\n');

}while (str != "/Ñèñòåìà\t");

}

allStatistic.push\_back(bet);

}

file.close();

}

}

void TForm6::CalculateStatistic(TDate date1, TDate date2)

{

TDate delta = date2 - date1;

double days = delta.Val;

if (days == 0)

{

days = 1;

}

int \_date1 = int(date1.Val);

int \_date2 = int(date2.Val);

int size = allStatistic.size();

int allBet = 0;

float balance = 0;

int winBet = 0;

list<OneBet> copyAllStatistic = allStatistic;

for (int i = 0; i < size; i++)

{

OneBet bet = copyAllStatistic.front();

copyAllStatistic.pop\_front();

TDate temp = StrToDate(bet.date.c\_str());

int \_temp = int(temp.Val);

if ((\_temp <= \_date2) && (\_temp >= \_date1))

{

allBet++;

balance += bet.win;

if (bet.win > 0)

{

winBet++;

}

}

}

if (allBet > 0)

{

balance = -balance;

Edit1->Text = FloatToStr(balance);

Edit2->Text = IntToStr(allBet);

double betInOneDay = allBet / days;

betInOneDay = int(betInOneDay\*100 + 0.5)/100.0;

Edit3->Text = FloatToStr(betInOneDay);

double procent = ((double(winBet) / allBet) \* 100);

Edit4->Text = FloatToStr(procent);

}

else

{

Edit1->Text = "0";

Edit2->Text = "0";

Edit3->Text = "0";

Edit4->Text = "0";

}

Label10->Caption = "Ñ " + DateToStr(date1) + " ïî " + DateToStr(date2);

}

void \_\_fastcall TForm6::FormCreate(TObject \*Sender)

{

Edit1->Clear();

Edit2->Clear();

Edit3->Clear();

Edit4->Clear();

string date2 = (DateToStr(DateTimePicker2->Date)).c\_str();

int pos = date2.find('.', 1);

string day = date2.substr(0, pos);

string monthAndYear = date2.substr(pos, date2.length() - 1);

int day1 = StrToInt(day.c\_str());

day1--;

string date1 = (FloatToStr(day1)).c\_str() + monthAndYear;

DateTimePicker1->Date = StrToDate(date1.c\_str());

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm6::DateTimePicker1Change(TObject \*Sender)

{

TDate date1 = DateTimePicker1->Date;

TDate date2 = DateTimePicker2->Date;

Form6->CalculateStatistic(date1, date2);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm6::DateTimePicker2Change(TObject \*Sender)

{

TDate date1 = DateTimePicker1->Date;

TDate date2 = DateTimePicker2->Date;

Form6->CalculateStatistic(date1, date2);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm6::Label4Click(TObject \*Sender)

{

TDate date1 = dateOfficeBirthDay;

TDate date2 = Date();

Form6->CalculateStatistic(date1, date2);

}

//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

#ifndef WorkWithFileH

#define WorkWithFileH

using namespace std;

struct Match

{

string data;

string time;

string home;

string guest;

int winnerHome;

int draw;

int winnerGuest;

};

list<Match> football;

list<Match> hockey;

list<Match> basketball;

void LoadSport(string fileName, list<Match>& sport);

list<Match> TodayMathes(list<Match> sport, string currencyData);

list<Match> ChoiceSport(TComboBox \*choiceSport);

//---------------------------------------------------------------------------

#endif

//---------------------------------------------------------------------------

#pragma hdrstop

#include <vcl.h>

#include <iostream>

#include <string>

#include <Windows.h>

#include <string.h>

#include <fstream>

#include <locale.h>

#include <list>

#include <cstring.h>

#include "WorkWithFile.h"

using namespace System;

//---------------------------------------------------------------------------

#pragma package(smart\_init)

void LoadSports()

{

string fileName = "Football.txt";

LoadSport(fileName, football);

fileName = "Hockey.txt";

LoadSport(fileName, hockey);

fileName = "Basketball.txt";

LoadSport(fileName, basketball);

}

void LoadSport(string fileName, list<Match>& sport)

{

ifstream file(fileName.c\_str());

if (file == NULL)

{

cout << "Âñå î÷åíü ïëîõî " << endl;

system("pause");

exit(0);

}

string str;

Match currencyMatch;

while(!file.eof())

{

getline(file, str, ' ');

if (str.empty() || str == "\n")

continue;

currencyMatch.data = str;

getline(file, str, ' ');

currencyMatch.time = str;

getline(file, str, ' ');

currencyMatch.home = str;

getline(file, str, ' ');

currencyMatch.guest = str;

getline(file, str, ' ');

currencyMatch.winnerHome = StrToFloat(str.c\_str());

getline(file, str, ' ');

currencyMatch.draw = StrToFloat(str.c\_str());

getline(file, str, '\n');

currencyMatch.winnerGuest = StrToFloat(str.c\_str());

sport.push\_back(currencyMatch);

}

}

list<Match> TodayMathes(list<Match> sport, string currencyData)

{

int size = sport.size();

list<Match> returnList;

for (int i = 0; i < size; i++)

{

Match temp = sport.front();

sport.pop\_front();

if (temp.data == currencyData)

{

returnList.push\_back(temp);

}

}

return returnList;

}

list<Match> ChoiceSport(TComboBox \*choiceSport)

{

switch(choiceSport->ItemIndex)

{

case 0:

return football;

case 1:

return hockey;

case 2:

return basketball;

default:

return football;

}

}

//---------------------------------------------------------------------------

#ifndef BettingH

#define BettingH

using namespace std;

struct Bet

{

string date;

string home;

string guest;

string resultMatch;

float coefficient;

float money;

string currency;

bool isBetTreatment;

float winning;

};

string emptyCoupon = "Êóïîí ñòàâîê ïóñò";

int defaultMoney = 0;

string defaultCurrency = "BLR";

typedef list<int> intList;

typedef list<Bet> List;

List ordinarList;

List expressList;

List systemList;

list<string> Parse(string pars);

list<float> CalculateOrdinar(list<float> koefList, float money);

list<float> CalculateExpress(list<float> koefList, float money);

list<float> CalculateSystem(list<float> koefList, float money, int dimension);

void SaveOrdinarToFile(List& betList, ofstream& file, string currency, float money);

void SaveExpressToFile(List& betList, ofstream& file, string currency, float money);

void SaveSystemToFile(List& betList, ofstream& file, string currency, float money, string dimension);

list<intList> overkill(int size, int dimension);

float CalculateCoef(list<float> koefList, list<intList> allCombination, int dimension);

float GetElementList(list<float> koefList, int number);

//---------------------------------------------------------------------------

#endif

//---------------------------------------------------------------------------

#pragma hdrstop

#include <vcl.h>

#include <iostream>

#include <string>

#include <Windows.h>

#include <string.h>

#include <fstream>

#include <locale.h>

#include <list>

#include <cstring.h>

#include "Betting.h"

using namespace std;

using namespace System;

//---------------------------------------------------------------------------

#pragma package(smart\_init)

void AddBetList(list<Bet>& betList, string resultMatch, float koef, string date, string time, string teamHome, string teamGuest)

{

Bet newBet;

newBet.date = date;

newBet.home = teamHome;

newBet.guest = teamGuest;

newBet.resultMatch = resultMatch;

newBet.coefficient = koef;

//ò.ê. ñòàâêà çàíåñåíà â ñïèñîê, íî íà íåå åùå íå ïîñòàâëåíû äåíüãè

//çàíîñèì â ïîëÿ çíà÷åíèÿ ïî óìîë÷àíèþ

newBet.money = defaultMoney;

newBet.currency = defaultCurrency;

newBet.isBetTreatment = false;

newBet.winning = 0;

betList.push\_back(newBet);

}

void DeleteBetList(list<Bet>& betList, string date, string time, string teamHome, string teamGuest, string resultMatch)

{

Bet deleteBet;

deleteBet.date = date;

deleteBet.home = teamHome;

deleteBet.guest = teamGuest;

deleteBet.resultMatch = resultMatch;

int size = betList.size();

list<Bet> copyBetList = betList;

Bet temp;

for (int i = 0; i < size; i++)

{

temp = copyBetList.front();

copyBetList.pop\_front();

if ((temp.date == deleteBet.date) && (temp.home == deleteBet.home) && (temp.guest == deleteBet.guest) && (temp.resultMatch == deleteBet.resultMatch))

{

break;

}

}

//ò.ê. íå ðàáîòàåò íîðìàëüíîå óäàëåíèå, òî î÷èùàåì ñïèñîê

//è ïåðåçàïèñûâåì, íå äîáàâëÿÿ óäàëÿåìûé ýëåìåíò

copyBetList.clear();

copyBetList = betList;

betList.clear();

for (int i = 0; i < size; i++)

{

temp = copyBetList.front();

copyBetList.pop\_front();

if ((temp.date == deleteBet.date) && (temp.home == deleteBet.home) && (temp.guest == deleteBet.guest) && (temp.resultMatch == deleteBet.resultMatch))

{

continue;

}

betList.push\_back(temp);

}

}

void SaveOrdinarToFile(List& betList, ofstream& file, string currency, float money)

{

int size = betList.size();

for (int i = 0; i < size; i++)

{

file << "Îðäèíàð" << "\t\n";

Bet temp = betList.front();

betList.pop\_front();

file << temp.date << "\t";

file << temp.home << " ";

file << temp.guest << " ";

file << temp.resultMatch << " ";

file << temp.coefficient << " ";

if ((money == defaultMoney) && (currency == defaultCurrency))

{

file << temp.money << " ";

file << temp.currency << " ";

}

else

{

file << (FloatToStr(money)).c\_str() << " ";

file << currency << " ";

}

file << temp.isBetTreatment << " ";

file << temp.winning << "\n";

}

}

void SaveExpressToFile(List& betList, ofstream& file, string currency, float money)

{

file << "Ýêñïðåññ" << "\t\n";

int size = betList.size();

for (int i = 0; i < size; i++)

{

Bet temp = betList.front();

betList.pop\_front();

file << temp.date << "\t";

file << temp.home << " ";

file << temp.guest << " ";

file << temp.resultMatch << " ";

file << temp.coefficient << " ";

if ((money == defaultMoney) && (currency == defaultCurrency))

{

file << temp.money << " ";

file << temp.currency << " ";

}

else

{

file << (FloatToStr(money)).c\_str() << " ";

file << currency << " ";

}

file << temp.isBetTreatment << " ";

file << temp.winning << "\n";

}

file << "/Ýêñïðåññ" << "\t\n";

}

void SaveSystemToFile(List& betList, ofstream& file, string currency, float money, string dimension)

{

file << "Ñèñòåìà" << "\t\n";

int size = betList.size();

file << dimension << "\n";

for (int i = 0; i < size; i++)

{

Bet temp = betList.front();

betList.pop\_front();

file << temp.date << "\t";

file << temp.home << " ";

file << temp.guest << " ";

file << temp.resultMatch << " ";

file << temp.coefficient << " ";

if ((money == defaultMoney) && (currency == defaultCurrency))

{

file << temp.money << " ";

file << temp.currency << " ";

}

else

{

file << (FloatToStr(money)).c\_str() << " ";

file << currency << " ";

}

file << temp.isBetTreatment << " ";

file << temp.winning << "\n";

}

file << "/Ñèñòåìà" << "\t\n";

}

list<float> AllCoef(TMemo \*Memo1, TComboBox \*choiceBet, float money, int dimension)

{

list<float> koefList;

list<float> returnList;

if (Memo1->Lines->Strings[0] != emptyCoupon.c\_str())

{

int numberMatches = Memo1->Lines->Count;

for (int i = 0; i < numberMatches; i++) //äîáàâëÿåì âñå êîýôôèöèåíòû â ñïèñîê

{

if (((i + 1) % 2) == 0) //â Memo â êàæäîé ÷åòíîé ñòðî÷êå åñòü êîýôôèöèåíòû

{

list<string> matchik = Parse((Memo1->Lines->Strings[i]).c\_str());

string opa = matchik.back().c\_str();

float koef = StrToFloat(opa.c\_str());

koefList.push\_back(koef);

}

}

switch(choiceBet->ItemIndex)

{

case 0:

returnList = CalculateOrdinar(koefList, money);

break;

case 1:

returnList = CalculateExpress(koefList, money);

break;

case 2:

returnList = CalculateSystem(koefList, money, dimension);

break;

}

}

else

{

returnList.push\_back(0);

returnList.push\_back(0);

returnList.push\_back(0);

}

return returnList;

}

list<float> CalculateOrdinar(list<float> koefList, float money)

{

int size = koefList.size();

float win = 0;

for (int i = 0; i < size; i++)

{

float koef = koefList.front();

koefList.pop\_front();

win += koef \* money;

}

float allMoney = money \* size;

list<float> resultBet;

float resultKoef = 1;

resultBet.push\_back(allMoney);

resultBet.push\_back(win);

resultBet.push\_back(resultKoef);

return resultBet;

}

list<float> CalculateExpress(list<float> koefList, float money)

{

int size = koefList.size();

float resultKoef = 1;

for (int i = 0; i < size; i++)

{

float koef = koefList.front();

koefList.pop\_front();

resultKoef \*= koef;

}

float win = resultKoef \* money;

float allMoney = money;

list<float> resultBet;

resultBet.push\_back(allMoney);

resultBet.push\_back(win);

resultBet.push\_back(resultKoef);

return resultBet;

}

list<float> CalculateSystem(list<float> koefList, float money, int dimension)

{

int size = koefList.size();

float moneyOnOneBet = money / size;

list<intList> allCombination = overkill(size, dimension); //ïîëó÷àåì âñå âîçìîæíûå êîìáèíàöèè

size = allCombination.size();

list<float> combinationCoefficients;

float winMoney = 0;;

for (int i = 0; i < size; i++)

{

intList oneCombination = allCombination.front();

allCombination.pop\_front();

float winOneBet = moneyOnOneBet;

for(int j = 0; j < dimension; j++)

{

int numberCoef = oneCombination.front();

oneCombination.pop\_front();

float coef = GetElementList(koefList, numberCoef);

winOneBet \*= coef;

}

winMoney += winOneBet;

}

list<float> resultBet;

resultBet.push\_back(money);

resultBet.push\_back(winMoney);

resultBet.push\_back(1);//ò.ê. îáùåãî êýôà ó ñòàâîê Ñèñòåìà íåòó. òî ïðîñòî äîáàâëÿåì â ñïèñîê çíà÷åíèå 1

return resultBet;

}

float GetElementList(list<float> koefList, int number)

{

float element = 0;

for(int i = 1; i <= number; i++)

{

element = koefList.front();

koefList.pop\_front();

}

return element;

}

list<intList> overkill(int size, int dimension)

{

const int a[7] = {1, 2, 3, 4, 5, 6, 7};

intList result;

list<intList> allCombination;

for(int i=1; i<(1<<size); ++i)

{

for(int j = 0; j<size; ++j)

{

if(i & (1<<j))

{

result.push\_back(a[j]);

}

}

if (int(result.size()) == dimension)

{

allCombination.push\_back(result);

}

result.clear();

}

return allCombination;

}

//âîçâðàùàåò íàçâàíèå êîìàíäû è êýô íà åå ïîáåäó

list<string> Parse(string pars)

{

string temp = "";

list<string> result;

int size = pars.size();

for (int i = 0; i < size; i++)

{

if (((pars[i] == ' ') || (pars[i] == '\t')) && (temp != " "))

{

result.push\_back(temp);

temp = "";

}

else if ((pars[i] != ' ') && (pars[i] != '\t'))

{

temp += pars[i];

}

}

if (temp != " ")

{

result.push\_back(temp);

temp = "";

}

return result;

}

//âñå âûáðàííûå ñòàâêè äîëæíû áûòü â îäíîì ñïèñêå

//åñëè âûáðàíû 2 ñòàâêè îðäèíàð, à ïîòîì áûëî ðåøåíî ïîñòàâèòü ýêñïðåññ

//òî èç ñïèñêà îðäèíàðíûõ ñòàâîê ñòàâêè ïåðåéäóò â ñïèñîê ýêñïðåññ-ñòàâîê

void AllInOneList(list<Bet>& inList, list<Bet>& outList1, list<Bet>& outList2)

{

if (outList1.size() != 0)

{

inList = outList1;

}

else if (outList2.size() != 0)

{

inList = outList2;

}

outList1.clear();

outList2.clear();

}

//---------------------------------------------------------------------------

#ifndef MadeBetH

#define MadeBetH

#include "Betting.h"

//List mOrdinarList;

//list<List> mExpressList;

struct MResult

{

string date;

string home;

string guest;

string result;

};

list <MResult> mBResult;

string LoadOrdinarMBet(ifstream& file, ofstream& fileCopy, string currencyData, TMemo \*Memo1, TMemo \*Memo2, TMemo \*Memo3);

string LoadExpressMBet(ifstream& file, ofstream& fileCopy, string currencyData, TMemo \*Memo1, TMemo \*Memo2, TMemo \*Memo3);

string LoadSystemMBet(ifstream& file, ofstream& fileCopy, string currencyData, TMemo \*Memo1, TMemo \*Memo2, TMemo \*Memo3);

void ToMemo(Bet bet, TMemo \*Memo);

void ExpressAndSystemToMemo(string forMemo, TMemo \*Memo);

//---------------------------------------------------------------------------

#endif

//---------------------------------------------------------------------------

#pragma hdrstop

#include <vcl.h>

#include <iostream>

#include <string>

#include <Windows.h>

#include <string.h>

#include <fstream>

#include <locale.h>

#include <list>

#include <cstring.h>

#include <cstdio>

#include "MadeBet.h"

using namespace std;

using namespace System;

//---------------------------------------------------------------------------

#pragma package(smart\_init)

void LoadResult()

{

ifstream file("Results.txt");

if (file == NULL)

{

cout << "Âñå î÷åíü ïëîõî " << endl;

system("pause");

exit(0);

}

string str;

while(!file.eof())

{

MResult newResult;

getline(file, str, ' ');

if (str.empty() || str == "\n")

continue;

newResult.date = str;

getline(file, str, ' ');

newResult.home = str;

getline(file, str, ' ');

newResult.guest = str;

getline(file, str, '\n');

newResult.result = str;

mBResult.push\_back(newResult);

}

}

//---------------------------------------------------------------------------

#ifndef MoneyH

#define MoneyH

using namespace std;

void LoadMoneyFromFile(string path);

void MoneyDefault();

struct Cash

{

float money;

string currency;

};

list<Cash> allMoney;

//---------------------------------------------------------------------------

#endif

//---------------------------------------------------------------------------

#pragma hdrstop

#include <vcl.h>

#include <iostream>

#include <string>

#include <Windows.h>

#include <string.h>

#include <fstream>

#include <locale.h>

#include <list>

#include <cstring.h>

#include "WorkWithFile.h"

#include "Money.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

void LoadMoneyFromFile(string path)

{

//ñîõðàíåíèå äåíåã ïåðåäåëàé, åïòà

ifstream file(path.c\_str());

string str;

while(!file.eof() && file != NULL)

{

getline(file, str, ' ');

if ((!str.empty()) && (str != "\n"))

{

Cash newMoney;

newMoney.money = StrToFloat(str.c\_str());

getline(file, str, '\n');

newMoney.currency = str;

allMoney.push\_back(newMoney);

}

}

}

void MoneyDefault()

{

Cash temp;

temp.money = 0;

temp.currency = "BLR";

allMoney.push\_back(temp);

temp.money = 0;

temp.currency = "RR";

allMoney.push\_back(temp);

temp.money = 0;

temp.currency = "USD";

allMoney.push\_back(temp);

temp.money = 0;

temp.currency = "EUR";

allMoney.push\_back(temp);

}